Exploring Data with yt

Britton Smith

Before we begin...

```
cd workshop/M83/
cp /home/guest008/workshop/yt_scripts/* .
```

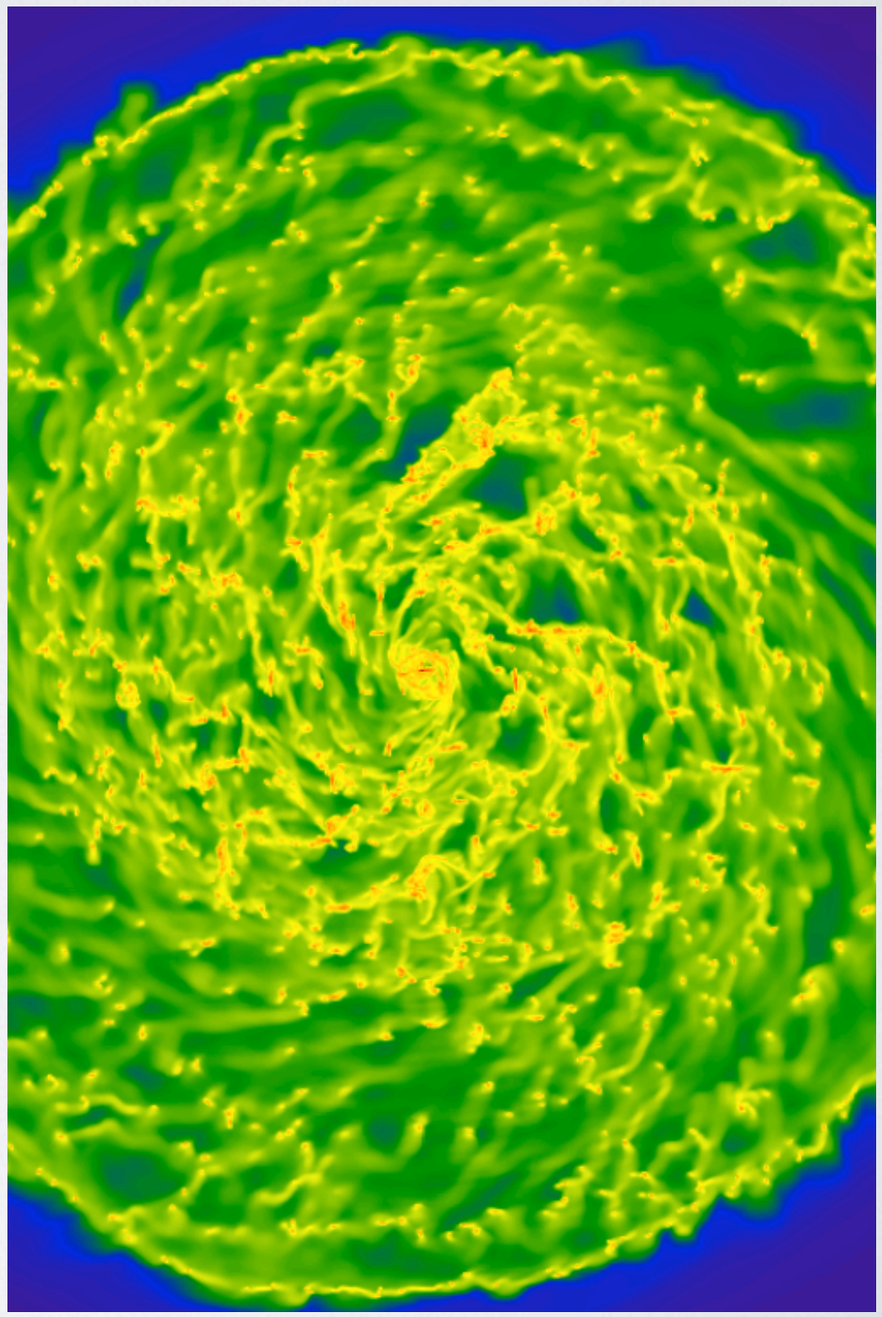
GOALS

1. Make plots

2. Work with data containers

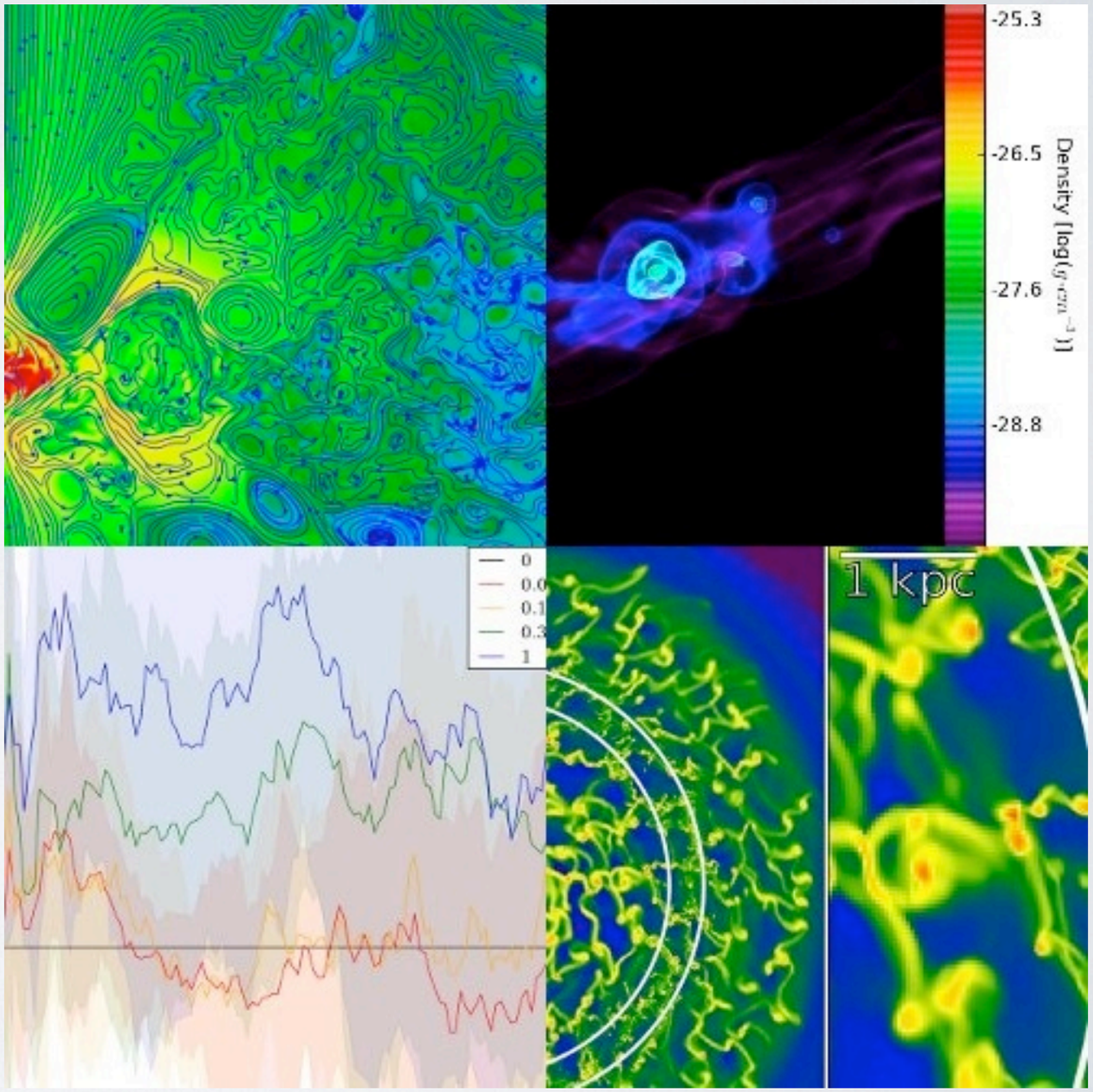
3. Make new fields

4. Advanced analysis: clump finding



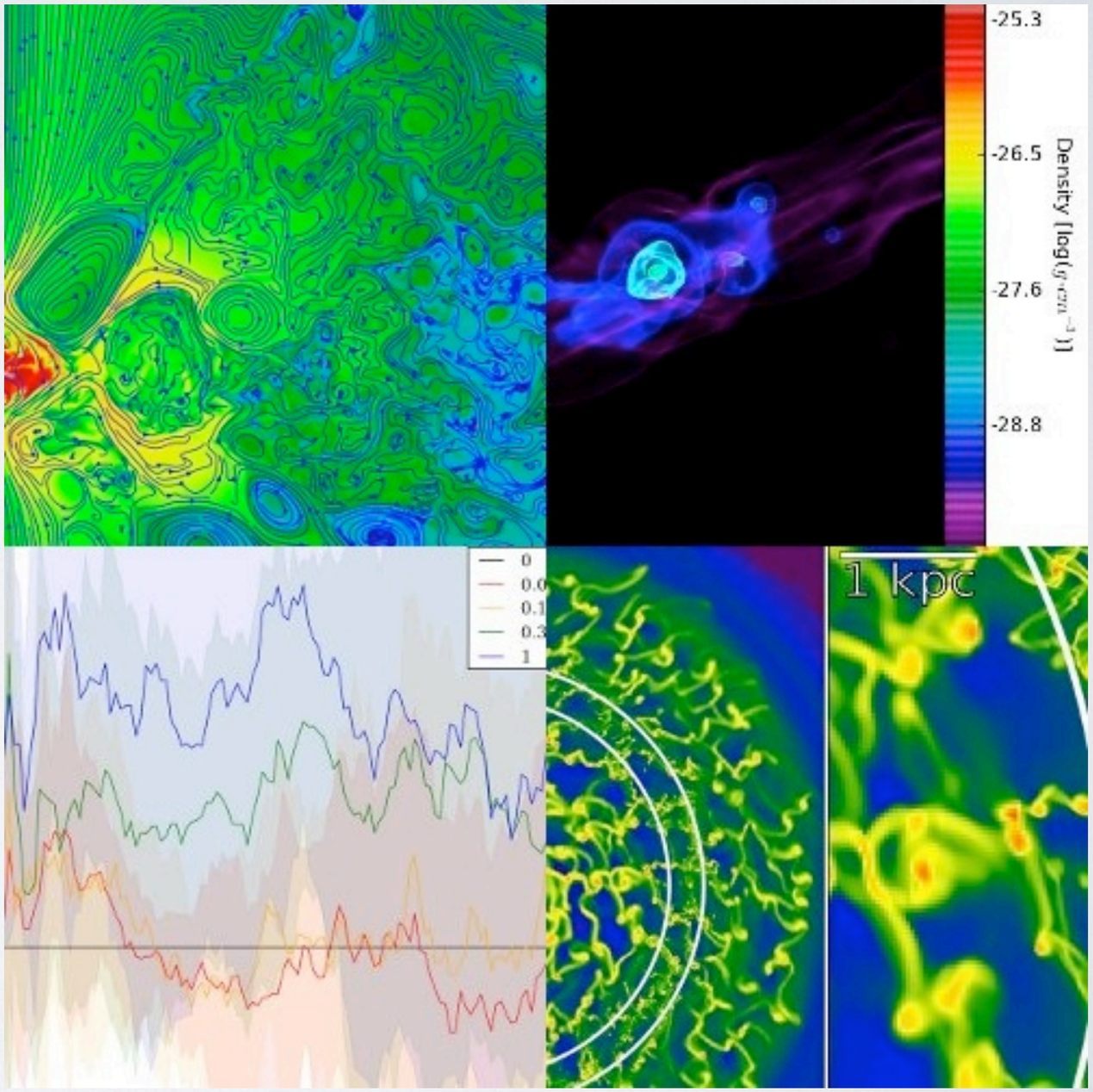
PLOTS

- Slices
- Projections
- Profiles
- Phase Plots

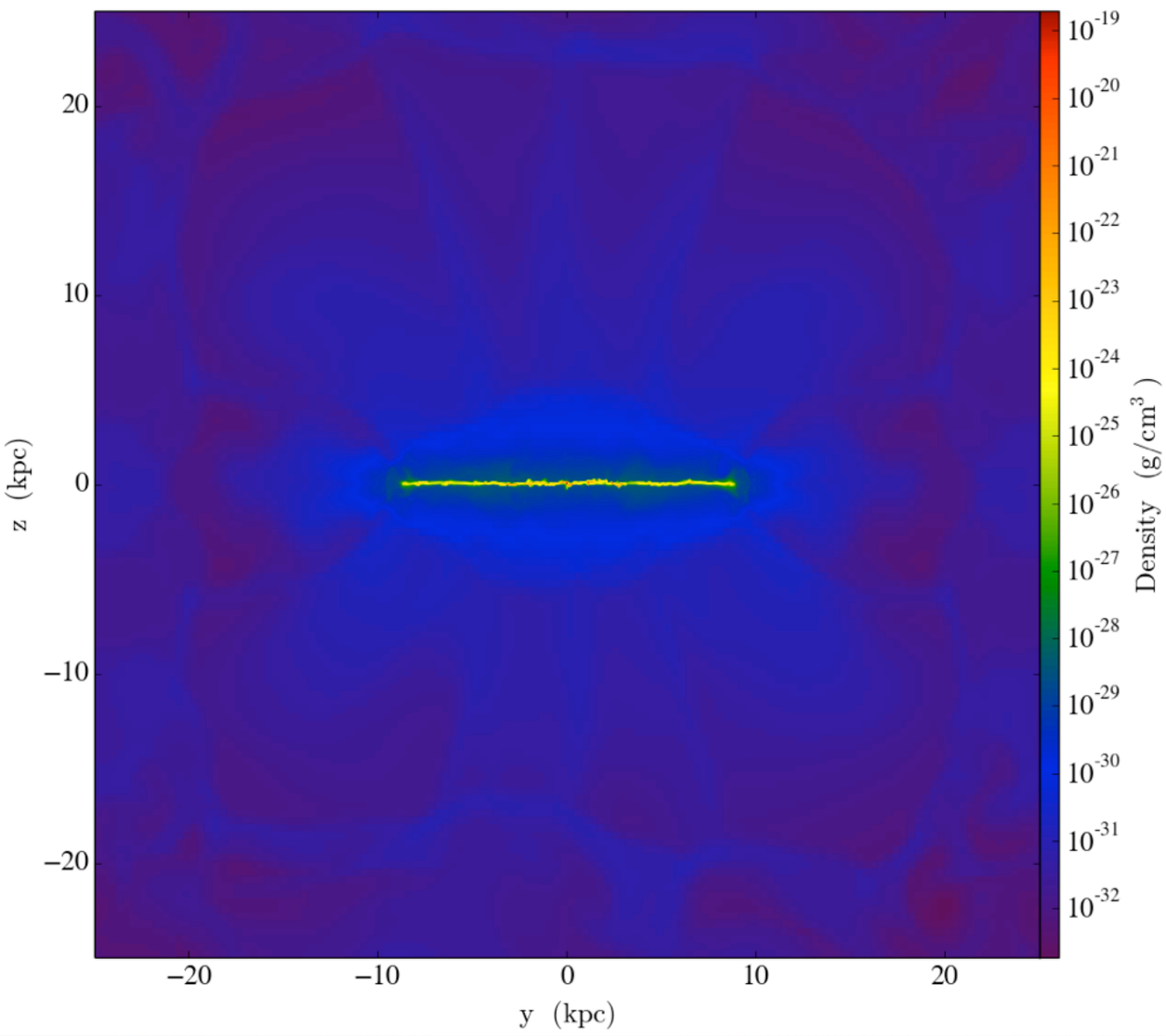


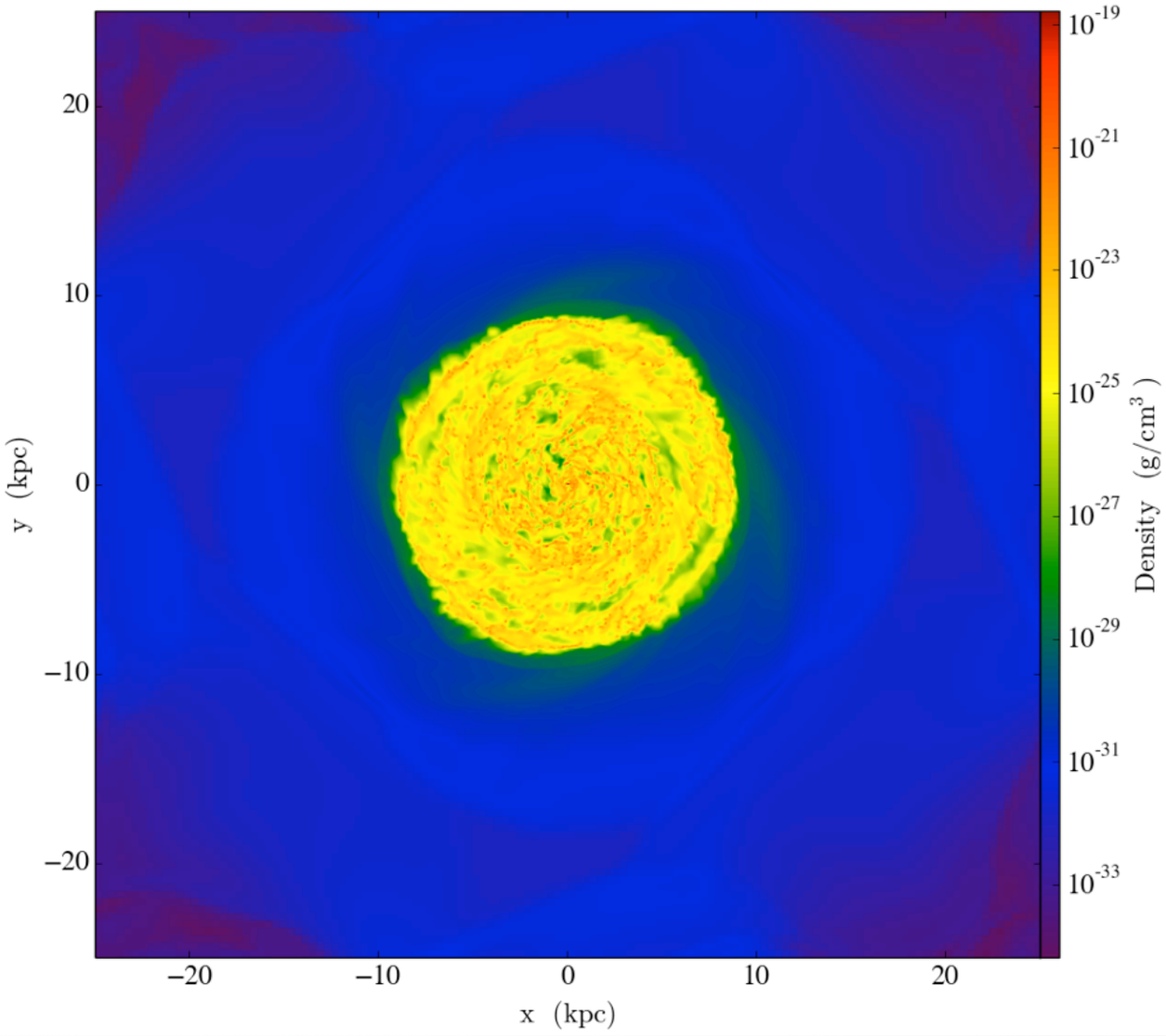
PLOTS

- Slices - try now
- Projections - try now
- Profiles - try later
- Phase Plots - try later

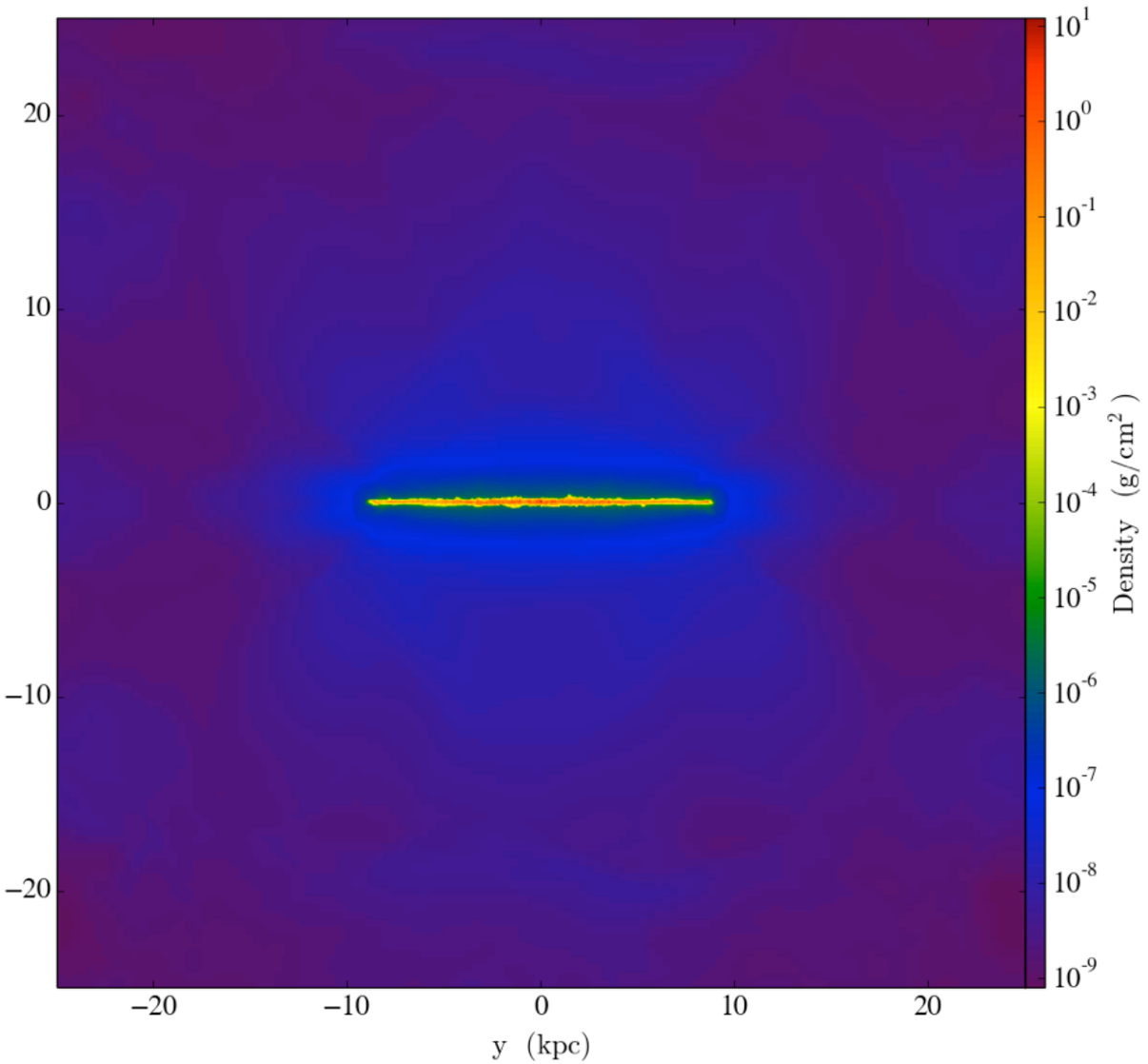


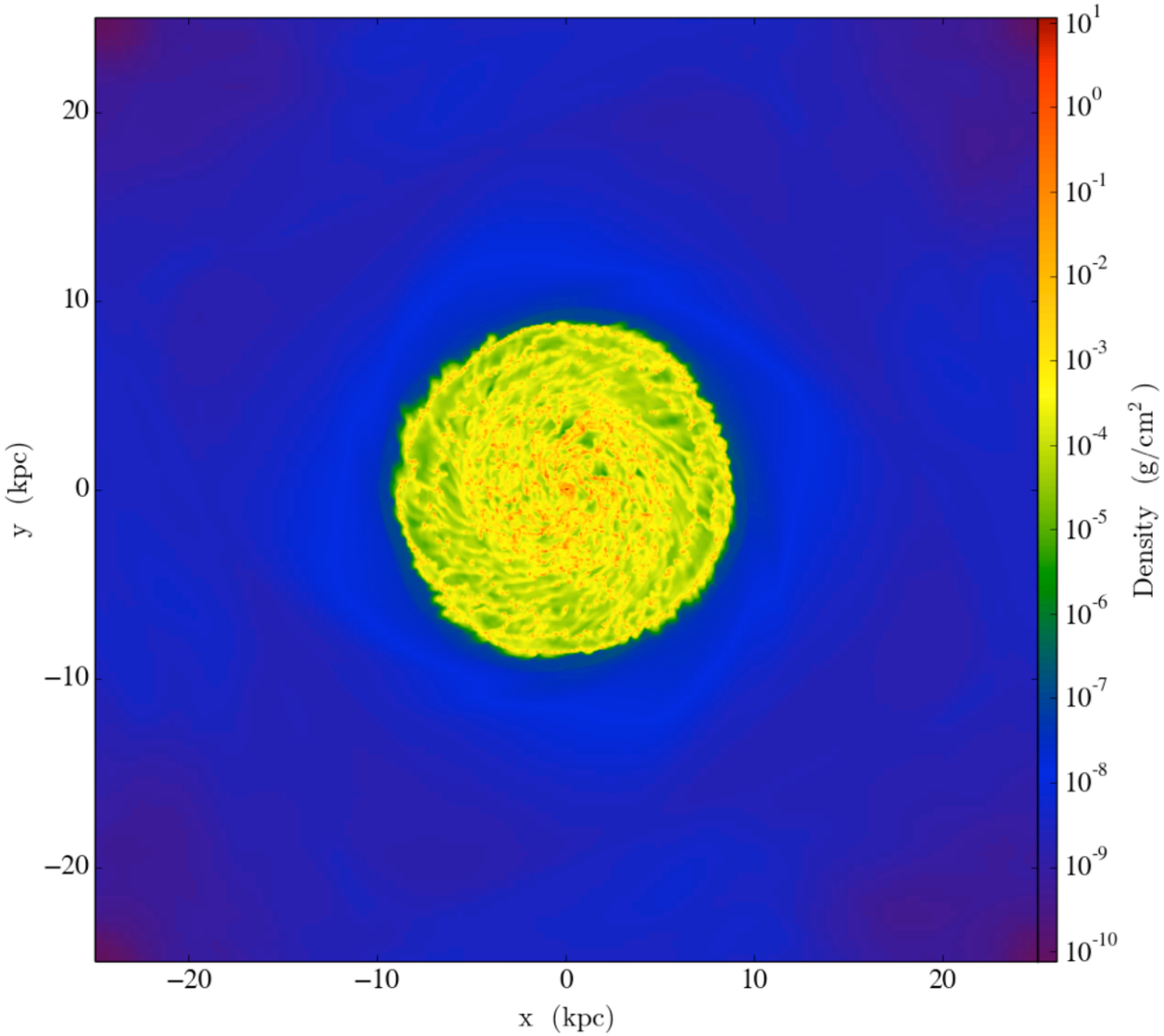
Slices - “01_slices.py”





Projections - “02_projections.py”

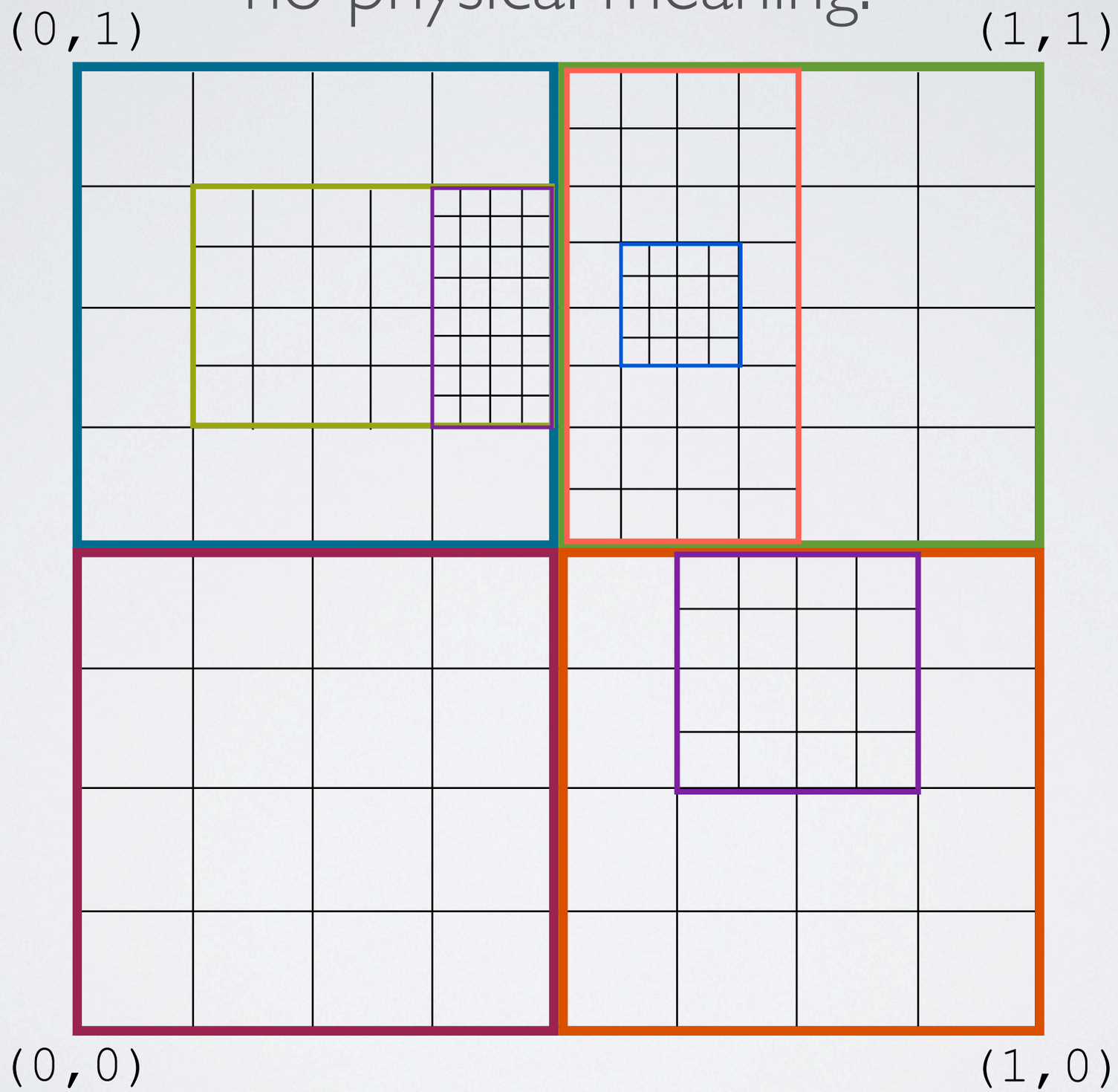




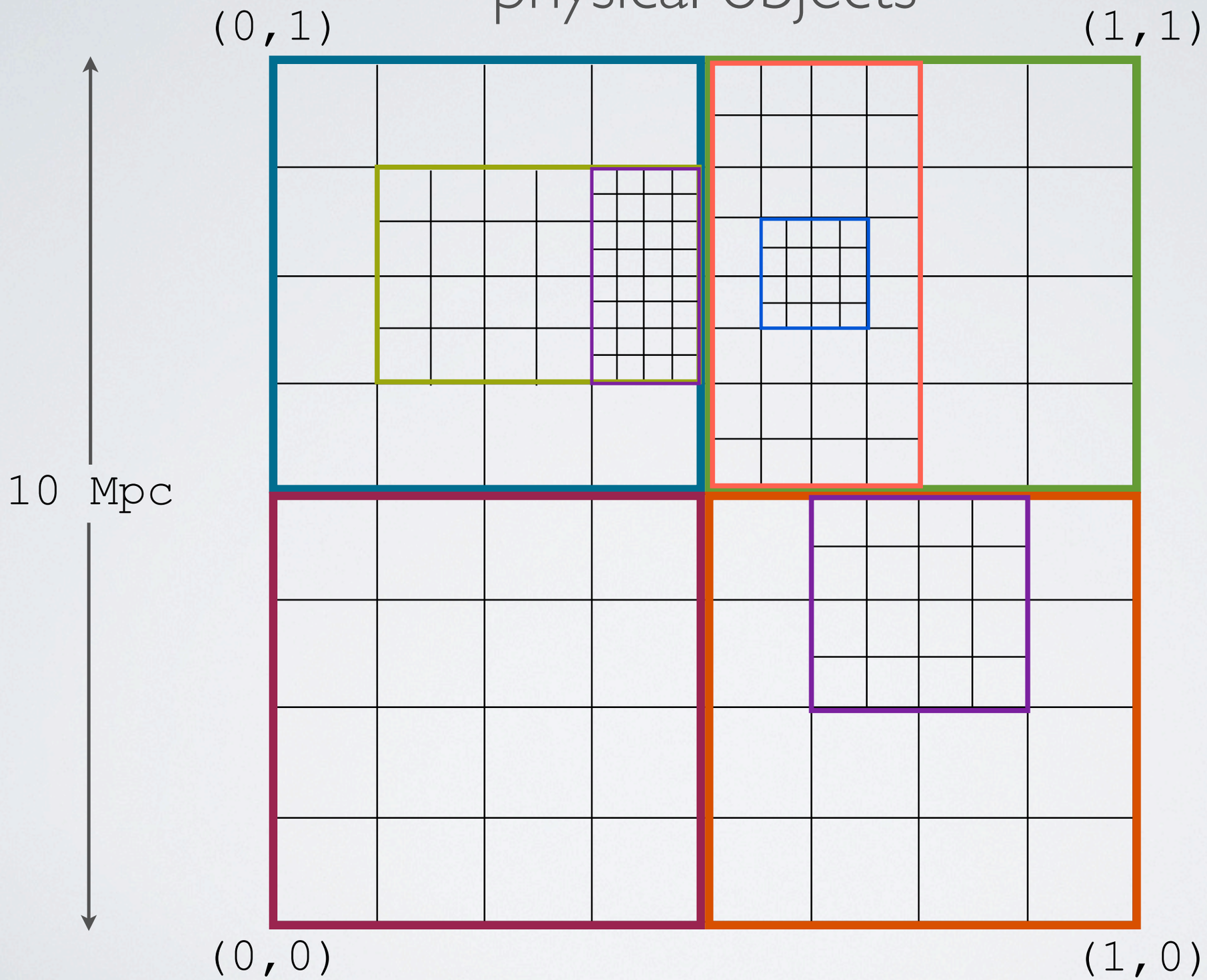
DATA CONTAINERS

yt gives you data
in "data containers."

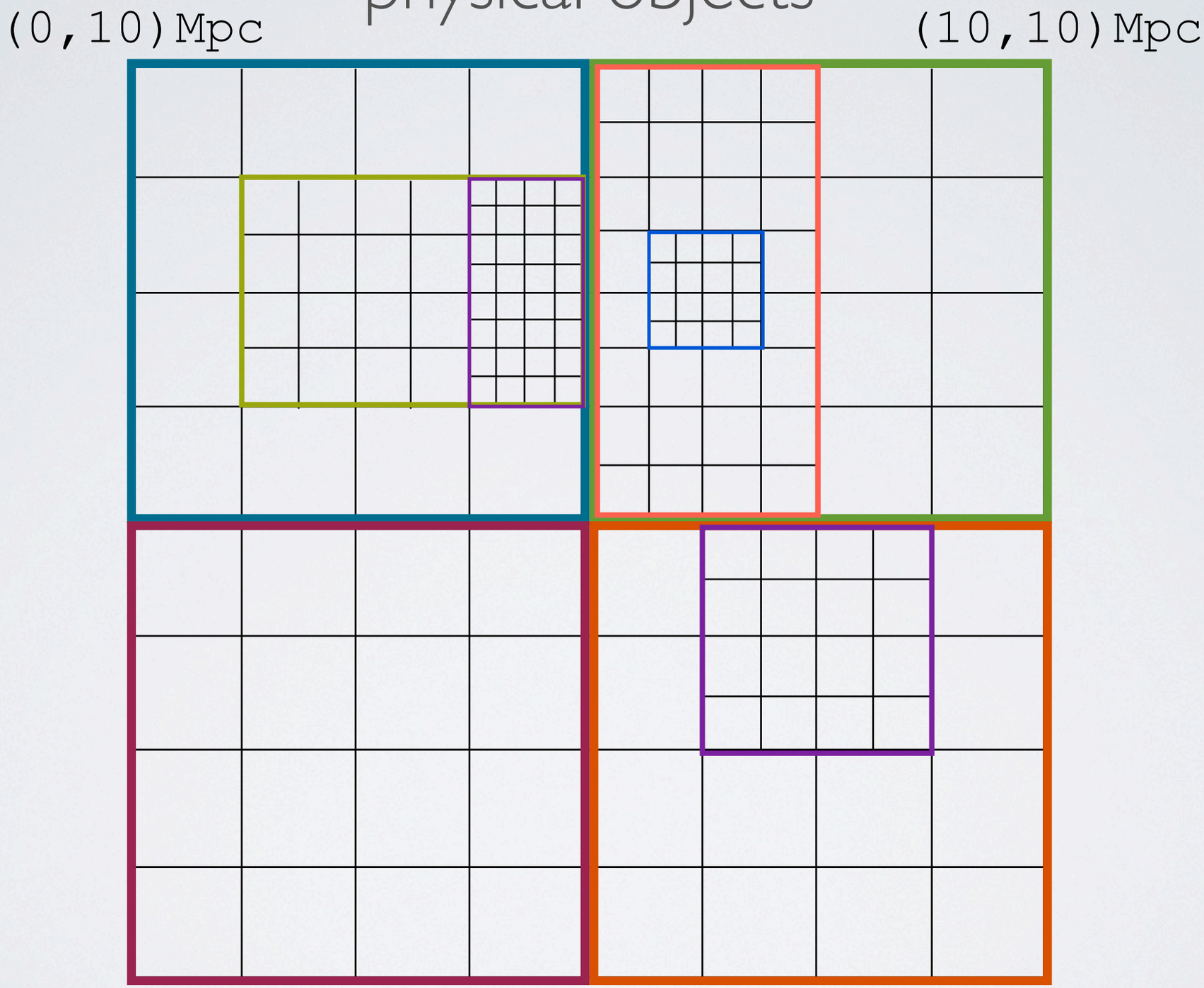
Data on disk has no physical meaning.



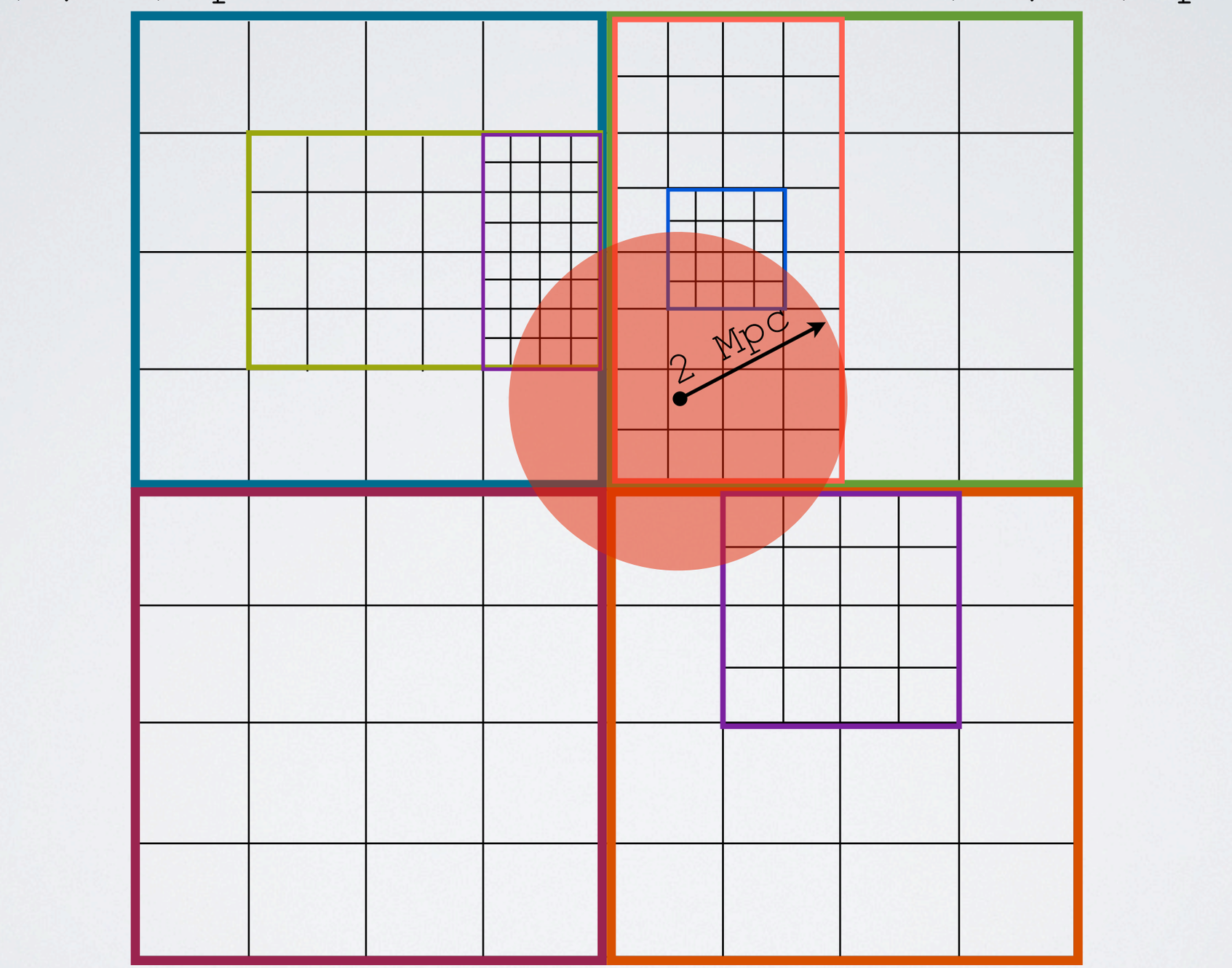
yt lets you think about physical objects



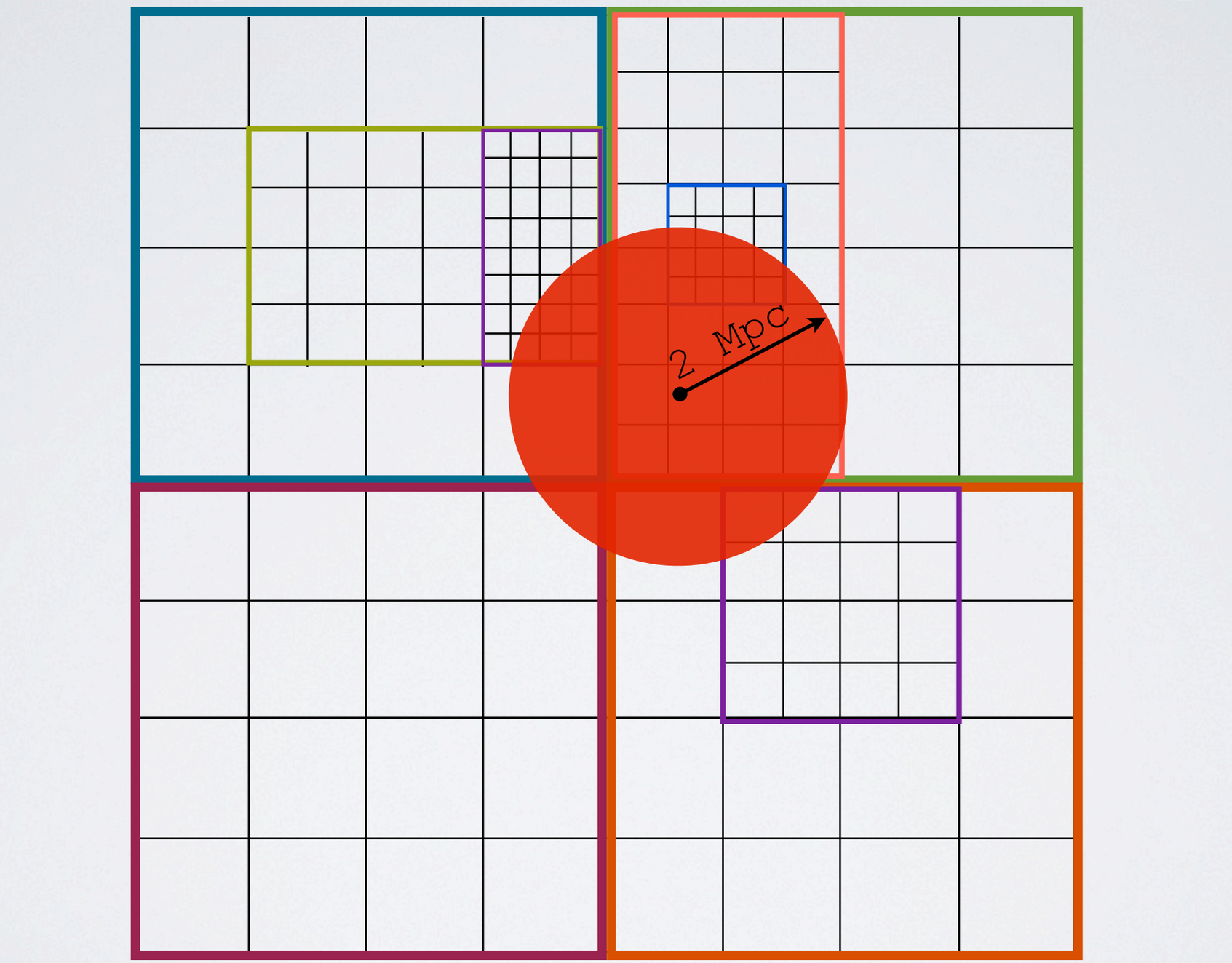
yt lets you think about physical objects



yt lets you think about physical objects

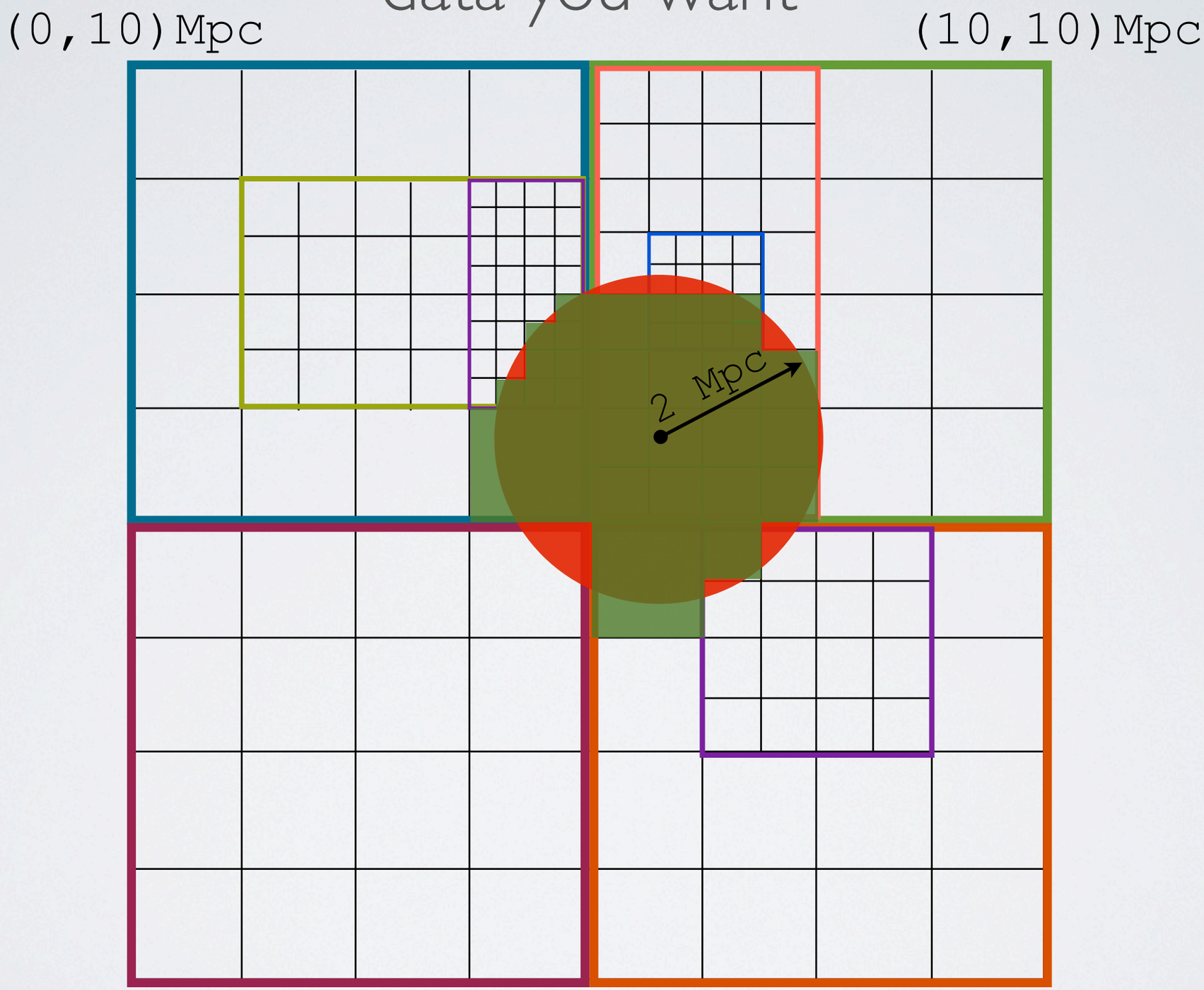


yt lets you think about physical objects

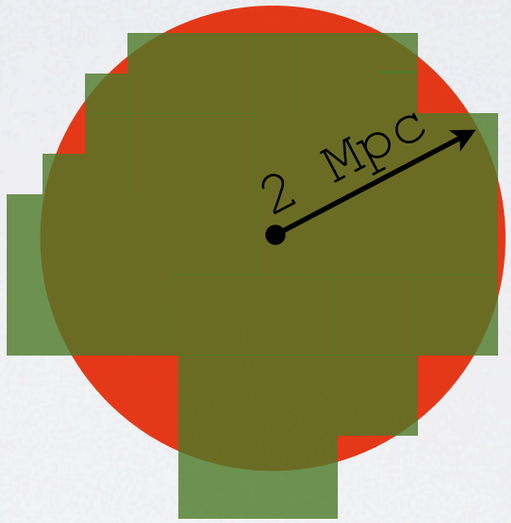


and forget what's underneath.

yt gives you the data you want

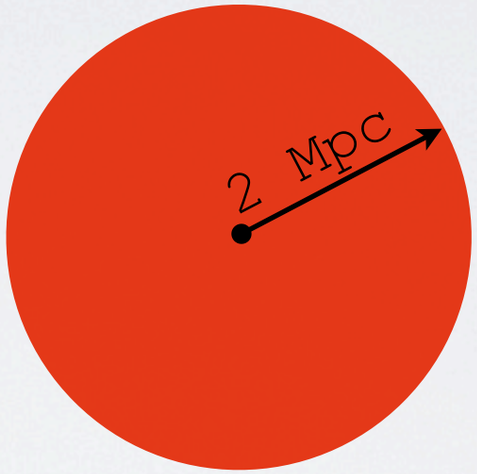


yt gives you the
data you want



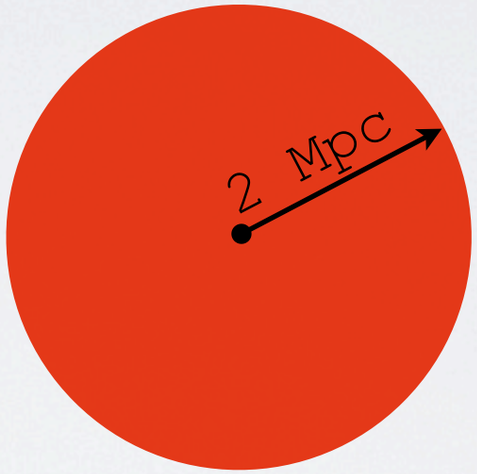
and only the data you want.

You can do whatever
you want with it.



You can do whatever you want with it.

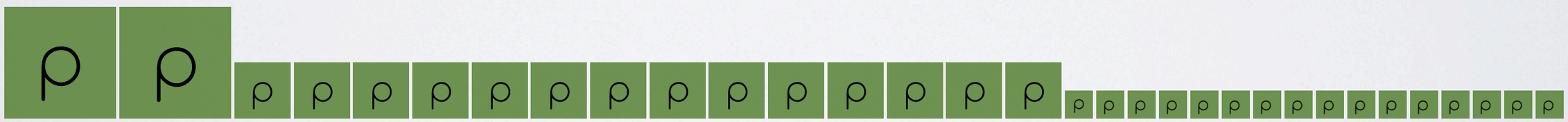
```
sp = pf.h.sphere(center,
                 (2, "mpc"))
```



You can do whatever you want with it.

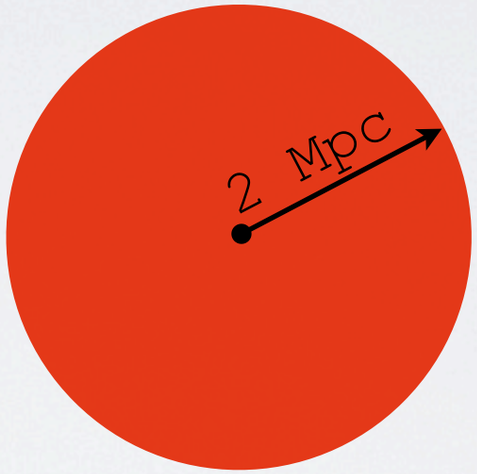
```
sp = pf.h.sphere(center,
                 (2, "mpc"))
```


```
sp["Density"]
```

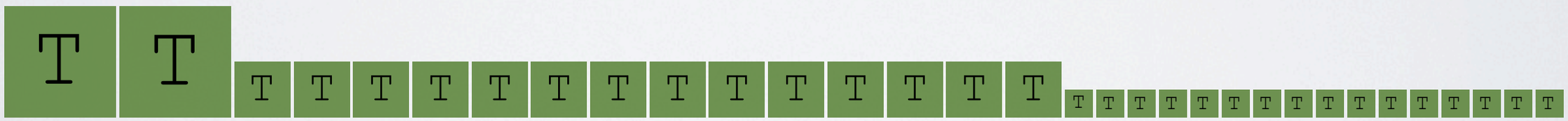


You can do whatever you want with it.

```
sp = pf.h.sphere(center,
                 (2, "mpc"))
```

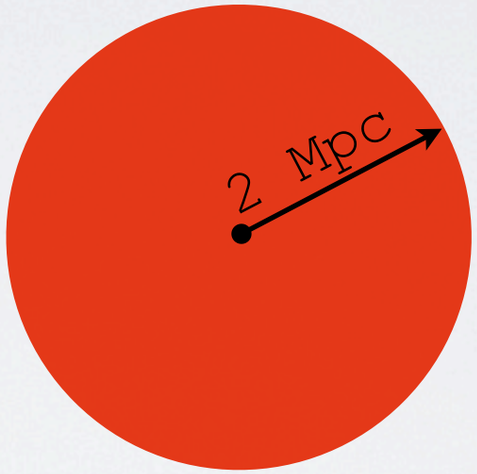



```
sp["Temperature"]
```

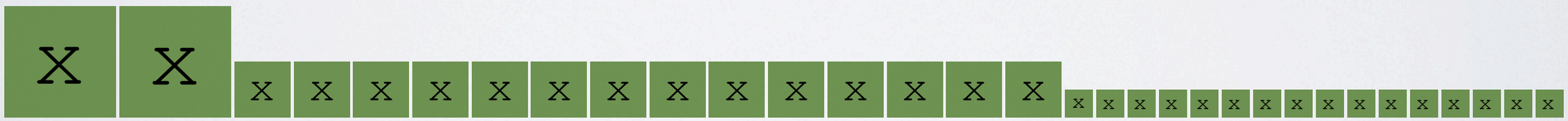


Position information is not lost.

```
sp = pf.h.sphere(center,
                 (2, "mpc"))
```



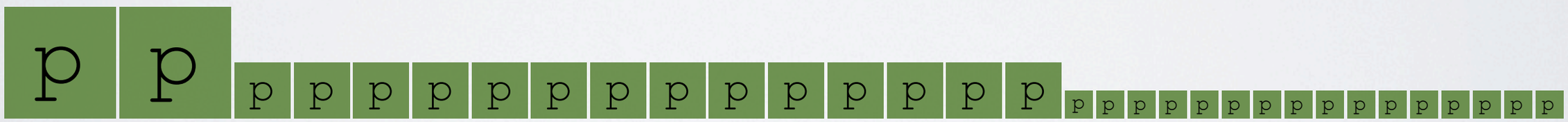
sp["x"]



Data containers give fields as Numpy arrays.

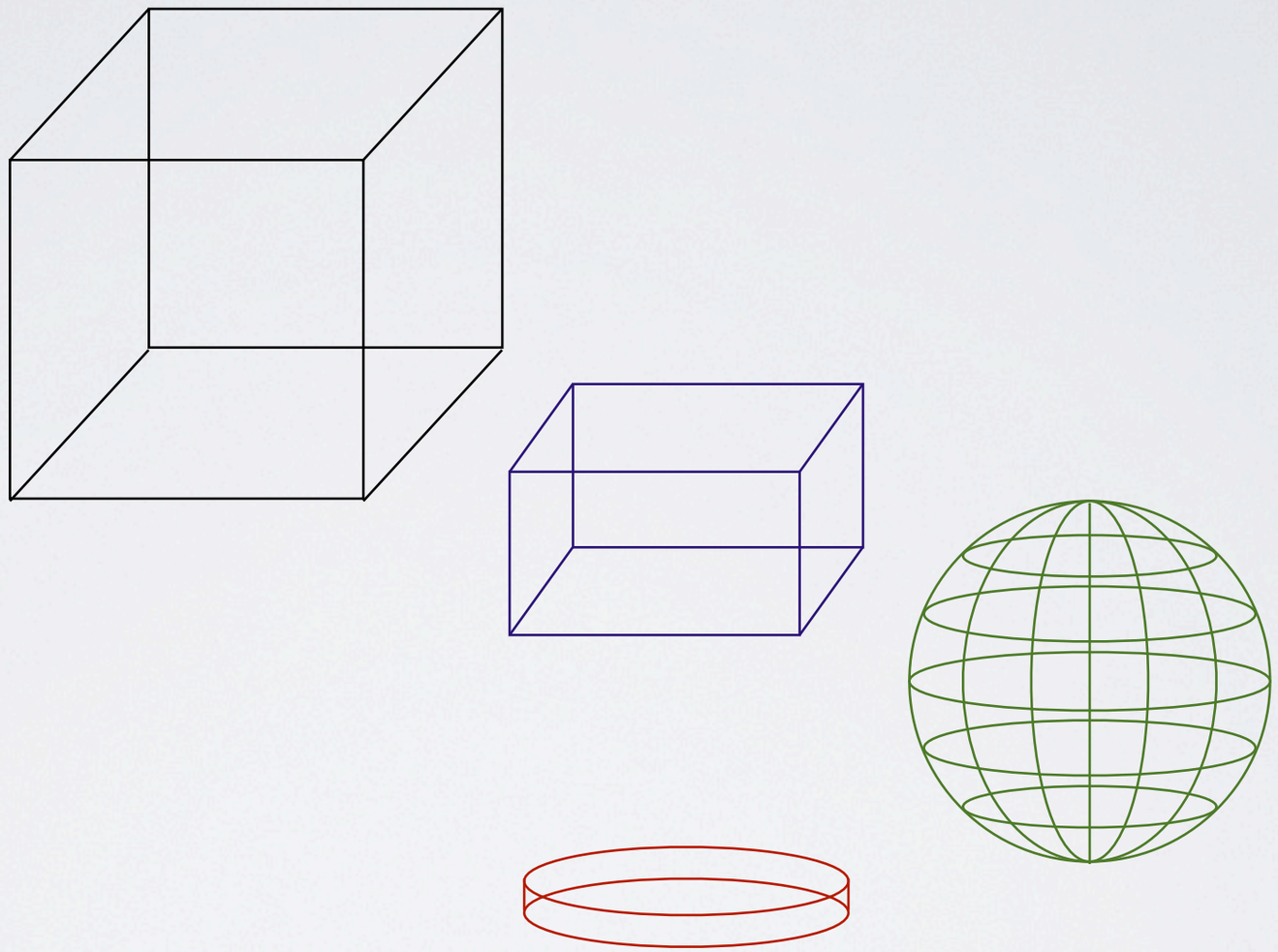
```
sp = pf.h.sphere(center,
                 (2, "mpc"))
```


```
sp["Density"] * sp["Temperature"]
```



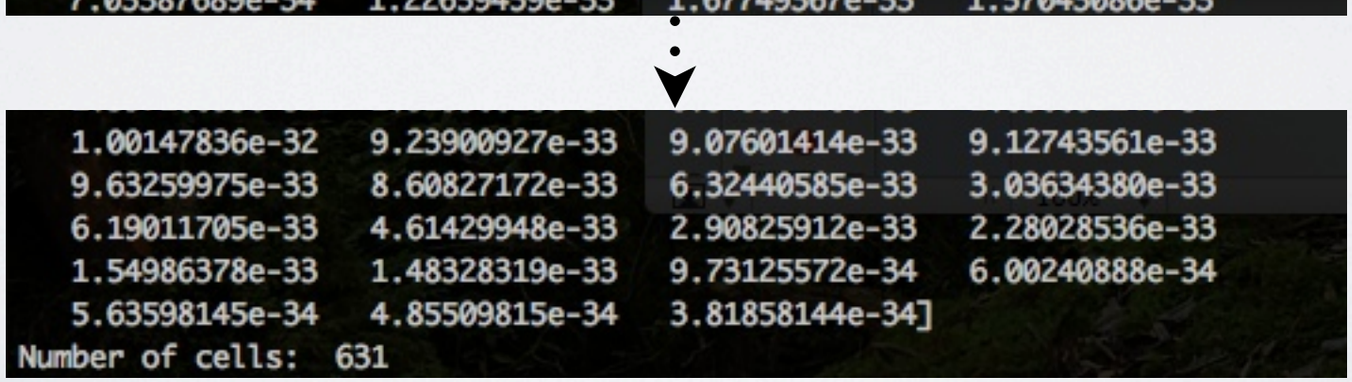
DATA CONTAINERS

- All Data
- Region
- Sphere
- Disk
- Ray



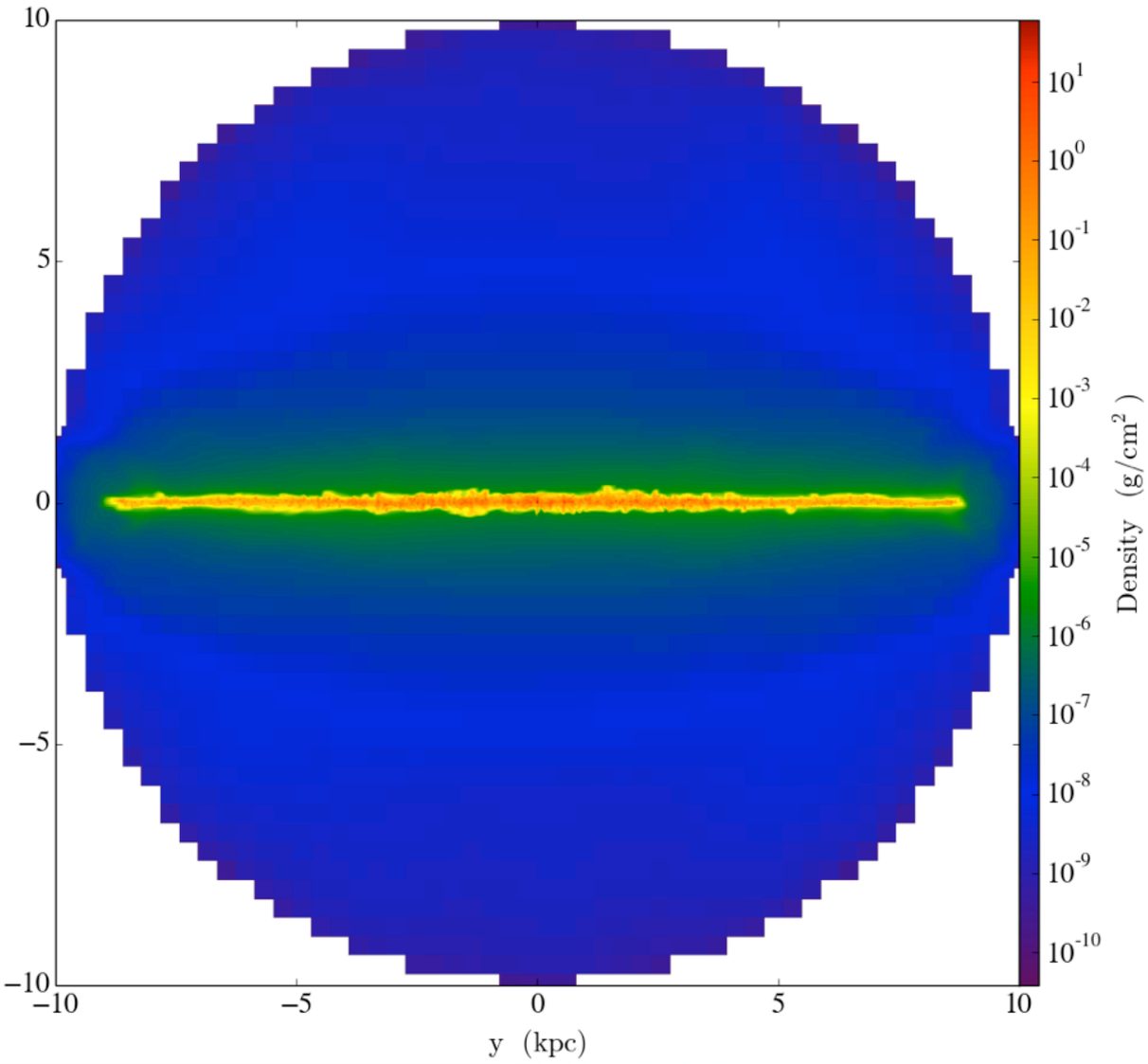
Data containers - “03_data_containers.py”

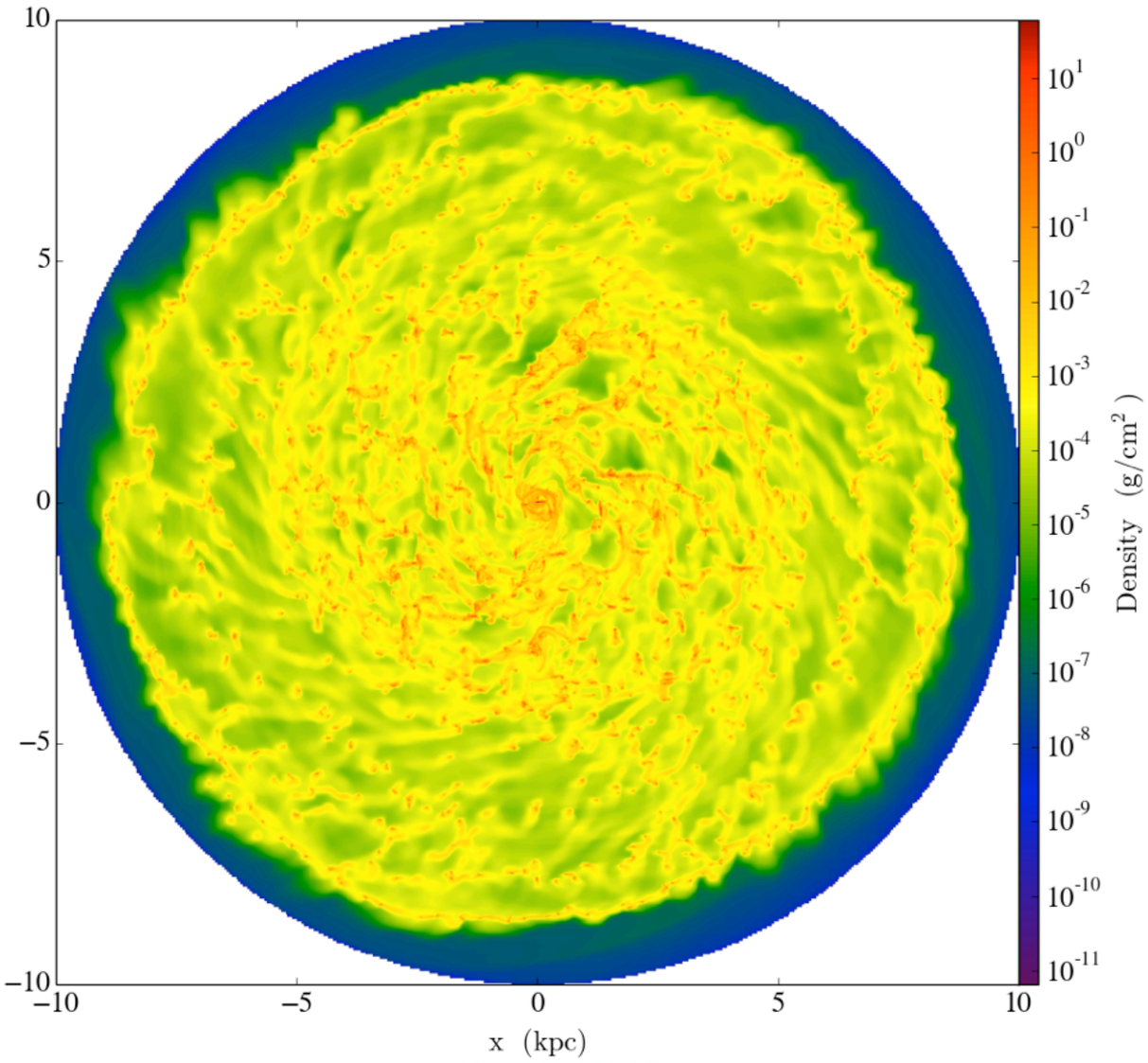
```
All data:
yt : [INFO     ] 2013-10-18 10:19:30,172 Getting field Density from 8714
[  4.38051338e-34   4.45548955e-34   4.45417363e-34 ...,   2.56533310e-31
   2.56964633e-31   2.57055874e-31]
Number of cells:  47003552
Region:
yt : [INFO     ] 2013-10-18 10:20:25,463 Getting field Density from 5695
[  1.89926932e-31   1.82609188e-31   1.77669009e-31 ...,   1.94573193e-27
   9.19540431e-26   1.32093682e-25]
Number of cells:  27519134
Sphere:
yt : [INFO     ] 2013-10-18 10:21:20,673 Getting field Density from 8564
[  4.62126245e-31   2.25309917e-31   5.25146029e-31 ...,   3.21442248e-24
   1.90200675e-24   1.45183160e-24]
Number of cells:  41508984
Disk:
yt : [INFO     ] 2013-10-18 10:24:17,551 Getting field Density from 8714
[  1.14037559e-25   1.52372888e-26   3.26771131e-27 ...,   2.42721258e-27
   1.65147053e-26   2.16783310e-25]
Number of cells:  41055352
Ray:
yt : [INFO     ] 2013-10-18 10:28:58,052 Getting field t from 62
yt : [INFO     ] 2013-10-18 10:28:58,156 Getting field Density from 62
[  4.38051338e-34   4.45548955e-34   6.30888871e-34   7.24134921e-34
   7.03387689e-34   1.22659459e-33   1.67749367e-33   1.57043086e-33
```



```
   1.00147836e-32   9.23900927e-33   9.07601414e-33   9.12743561e-33
   9.63259975e-33   8.60827172e-33   6.32440585e-33   3.03634380e-33
   6.19011705e-33   4.61429948e-33   2.90825912e-33   2.28028536e-33
   1.54986378e-33   1.48328319e-33   9.73125572e-34   6.00240888e-34
   5.63598145e-34   4.85509815e-34   3.81858144e-34]
Number of cells:  631
```

Sphere projections - “04_sphere_projection.py”

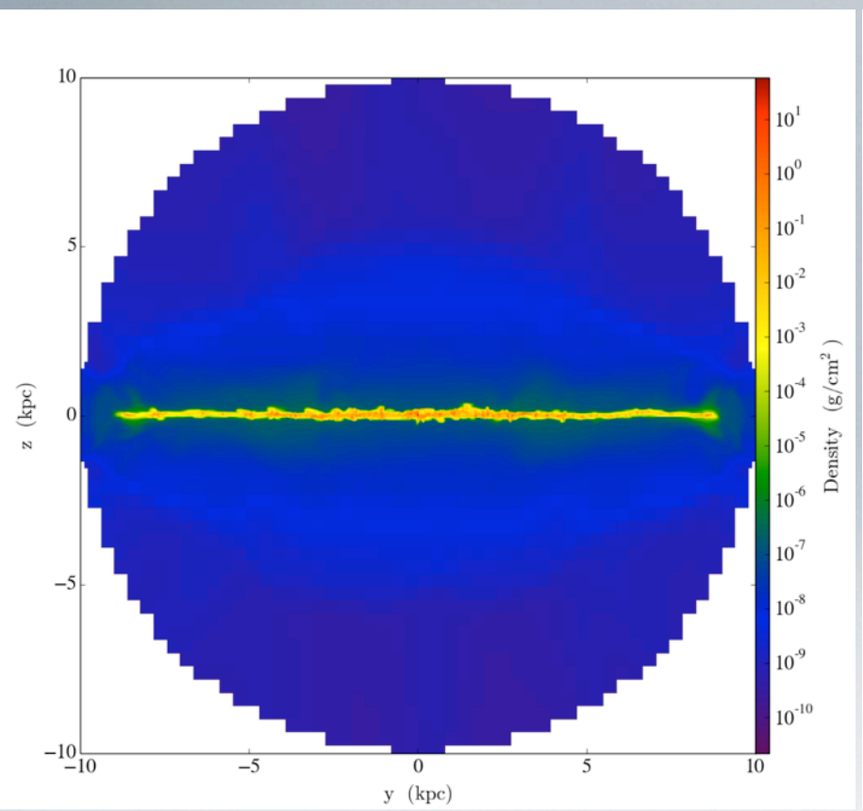


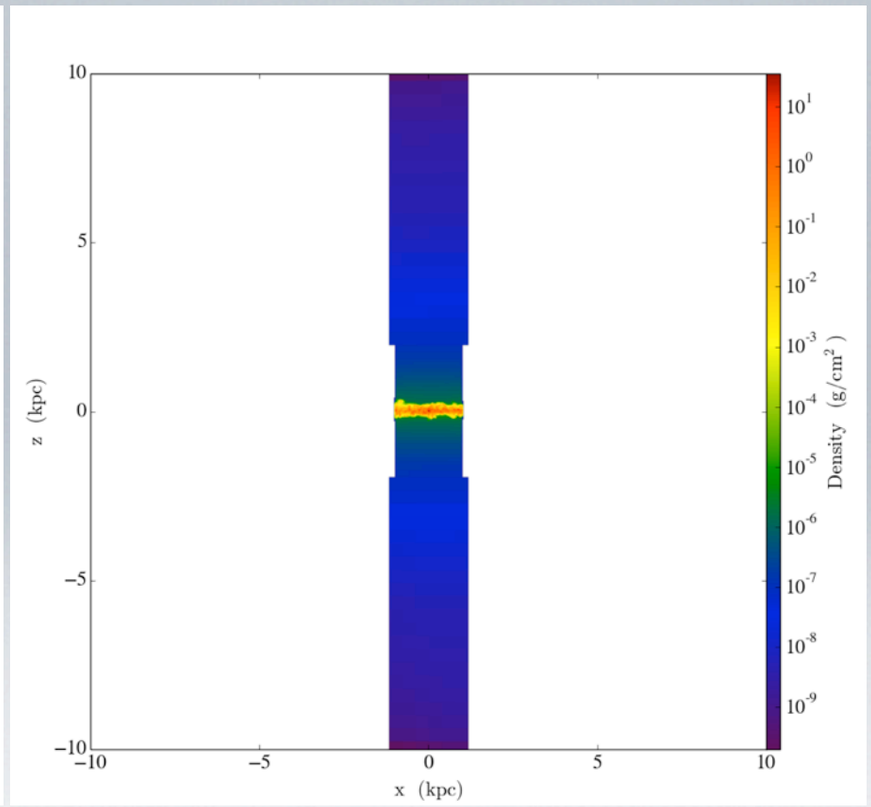


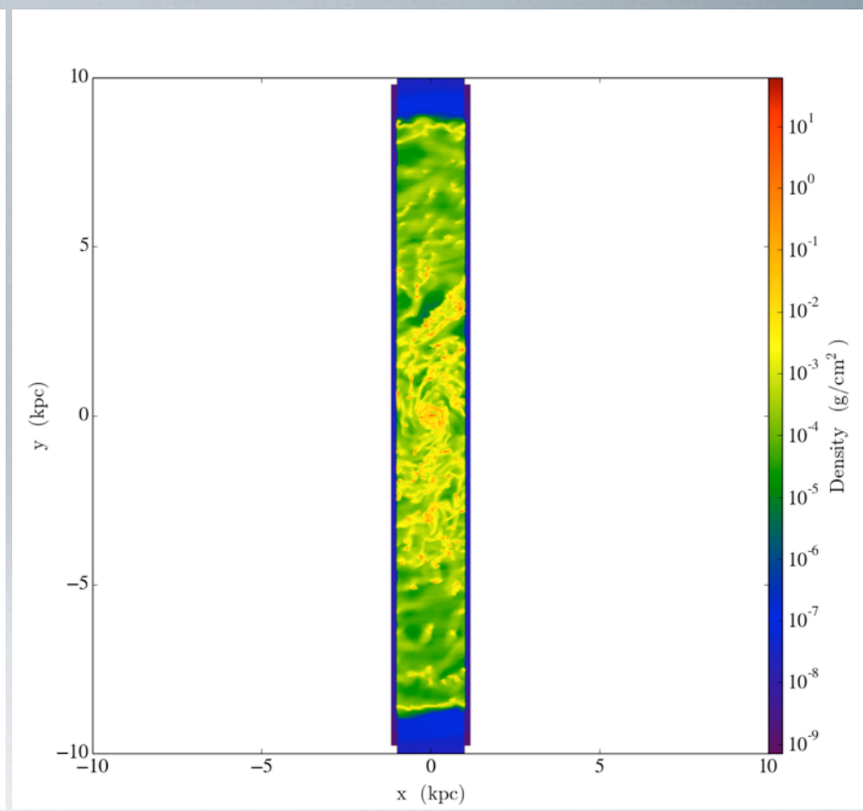
Disk projections - “05_disk_projection.py”

```
from yt.mods import *

pf = load("DD0200/R7_YC_0200")

sphere = pf.h.sphere(pf.domain_center, (10, 'kpc'))

normal = [1.0, 0, 0]
disk = pf.h.disk(pf.domain_center, normal,
                 (10, 'kpc'), (1, 'kpc'))

for axis in "xyz":
    plot = ProjectionPlot(pf, axis, "Density", width=(20, 'kpc'),
                          data_source=disk)
    plot.save(name="first_disk")
```

Disk projections - “05_disk_projection.py”





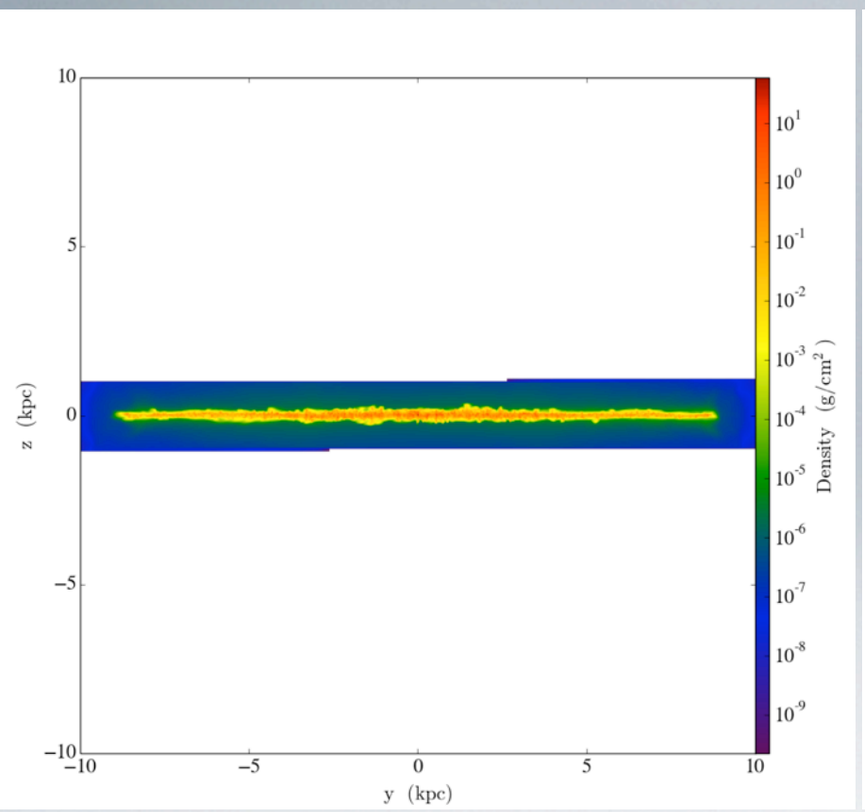


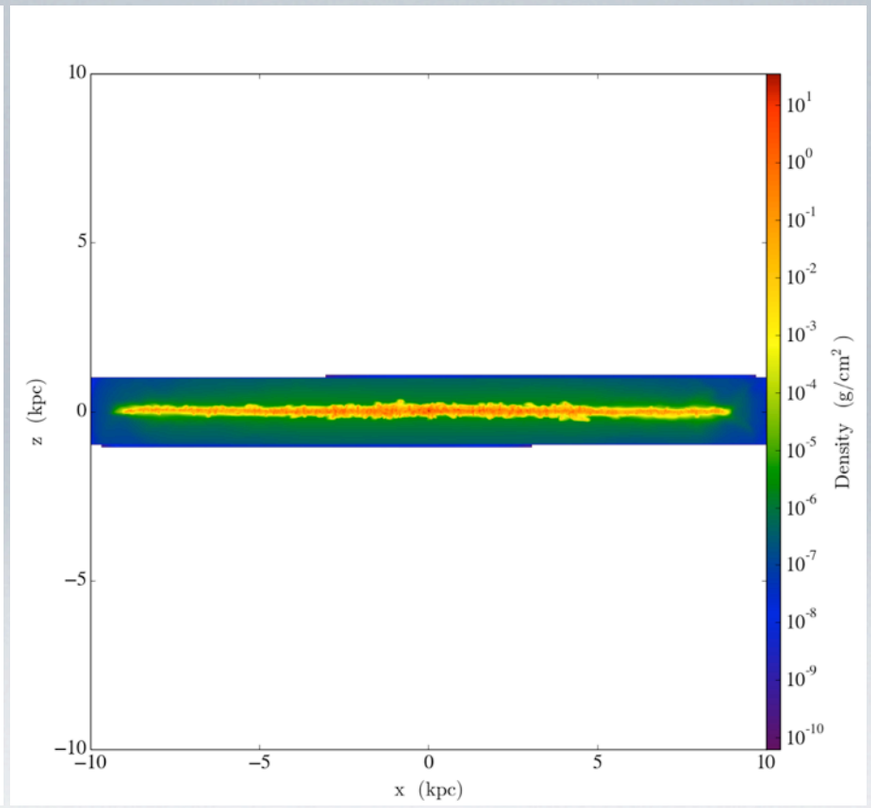
Uh oh!

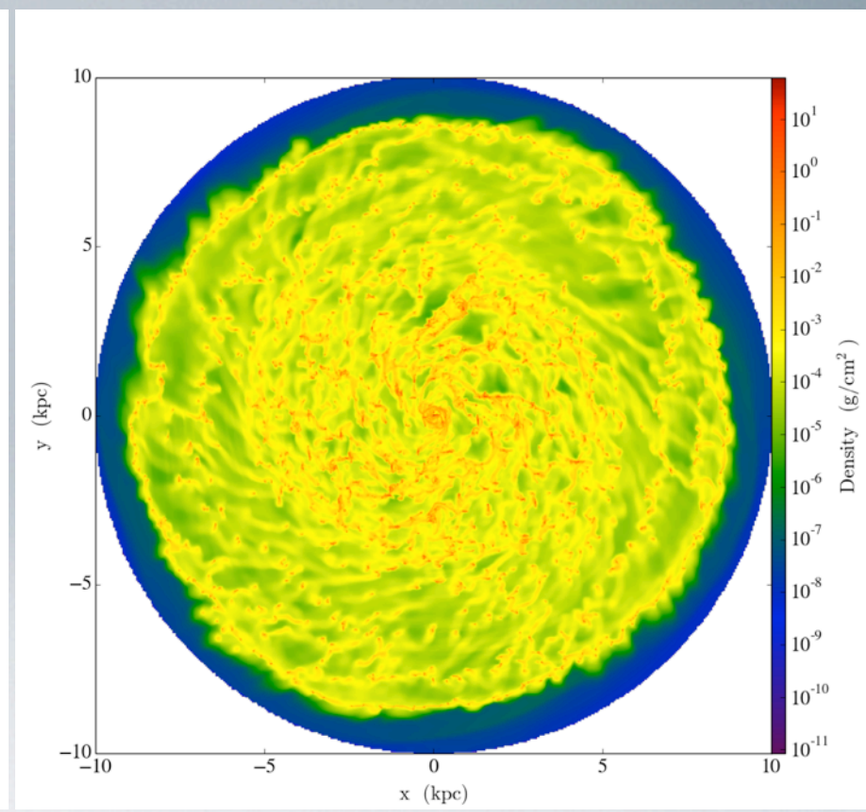
Disk projections - “06_disk_projection_2.py”

```
from yt.mods import *

pf = load("DD0200/R7_YC_0200")

sphere = pf.h.sphere(pf.domain_center, (10, 'kpc'))

J_vector = sphere.quantities['AngularMomentumVector']()
disk = pf.h.disk(pf.domain_center, J_vector,
                 (10, 'kpc'), (1, 'kpc'))

for axis in "xyz":
    plot = ProjectionPlot(pf, axis, "Density", width=(20, 'kpc'),
                          data_source=disk)
    plot.save(name="second_disk")
```

Disk projections - “06_disk_projection_2.py”



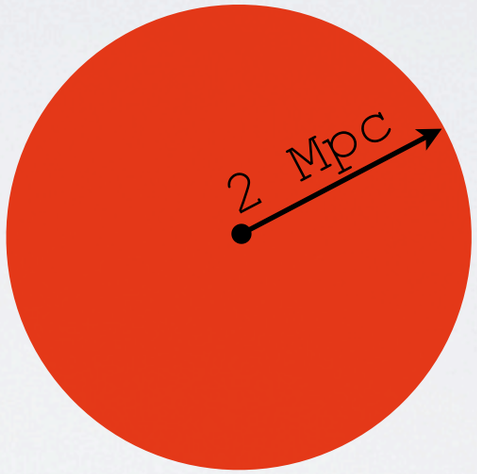




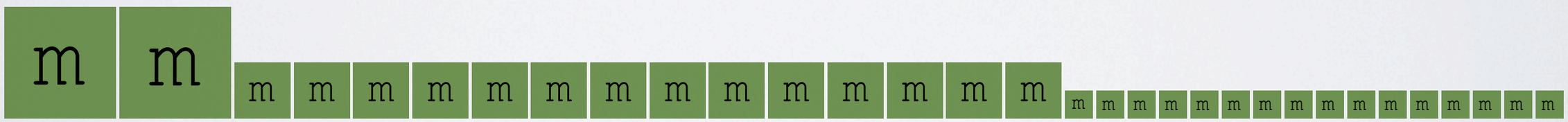
Much better!

Derived quantities turn fields into single values.

```
sp = pf.h.sphere(center,
                 (2, "mpc"))
```

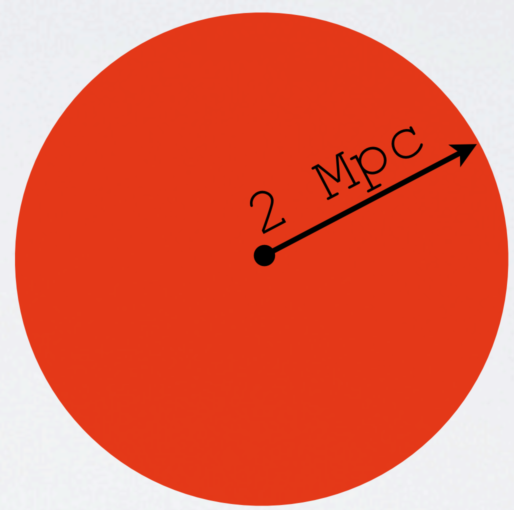



```
sp["CellMassMsun"]
```

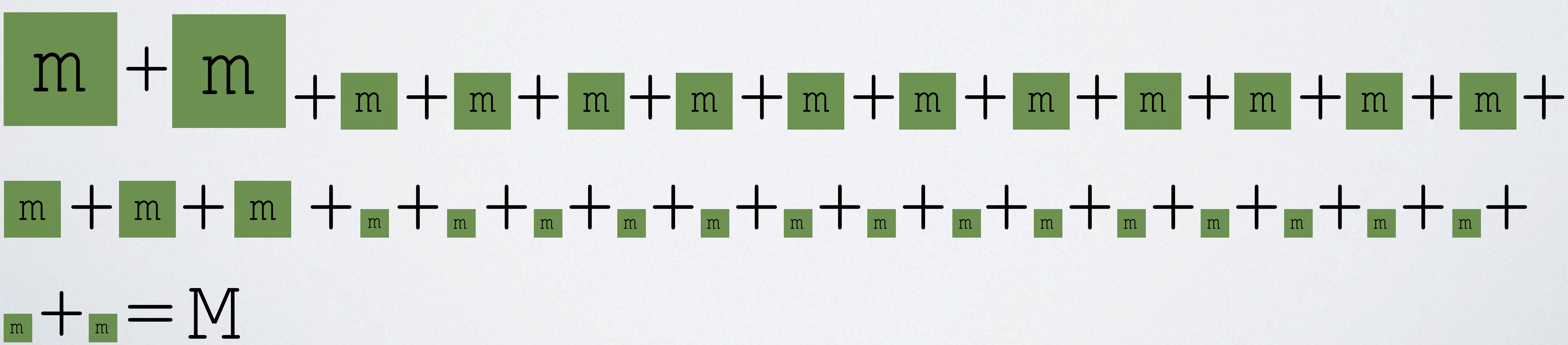


Derived quantities turn fields into single values.

```
sp = pf.h.sphere(center,
                 (2, "mpc"))
```

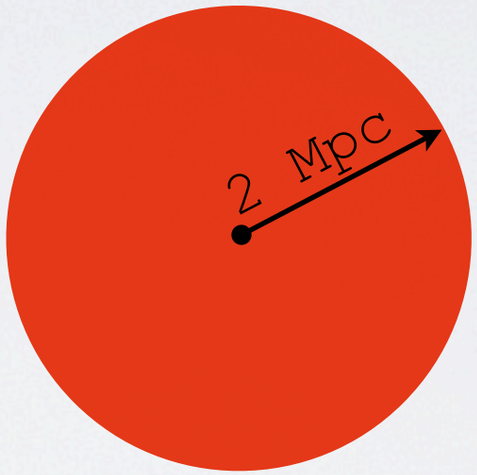



```
sp.quantities["TotalQuantity"]("CellMassMsun")
```



Derived quantities turn fields into single values.

```
sp = pf.h.sphere(center,
                 (2, “mpc”))
```




```
sp.quantities[“TotalQuantity”](“CellMassMsun”)
```

$$M = \sum m_i$$

Derived quantities turn fields into single values.

```
sp = pf.h.sphere(center,
                 (2, “mpc”))
```


```
sp.quantities[“WeightedAverageQuantity”](“Density”,
                                         “CellMassMsun”)
```

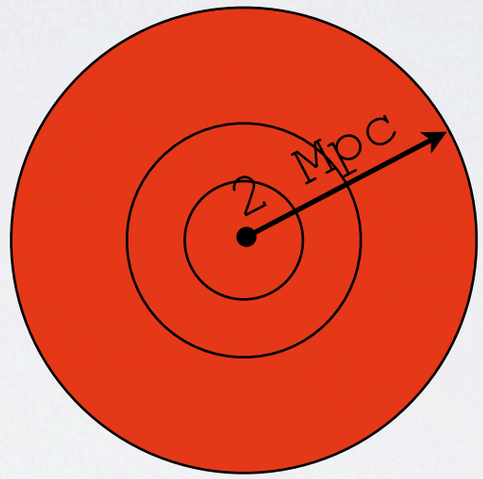
$$M = \frac{\sum \rho_i m_i}{\sum m_i}$$

Derived quantities - “07_derived_quantities.py”

```
from yt.mods import *

pf = load("DD0200/R7_YC_0200")

# make an "all data" container
my_all_data = pf.h.all_data()
print "All data:"
total_mass = my_all_data.quantities["TotalQuantity"]("CellMassMsun")
print "Total mass:", total_mass, "Msun"
```

Profiles allow you to plot one field versus another.

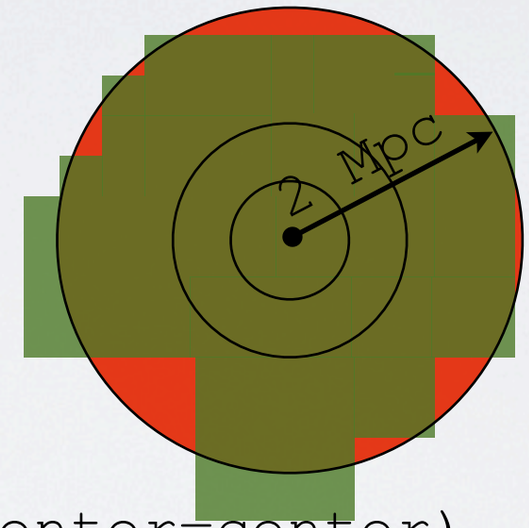
```
sp = pf.h.sphere(center,
                 (2, “mpc”))
```




```
pc = PlotCollection(pf, center=center)
pc.add_profile_object(sp, [“RadiusMpc”, “Density”])
```

Profiles allow you to plot one field versus another.

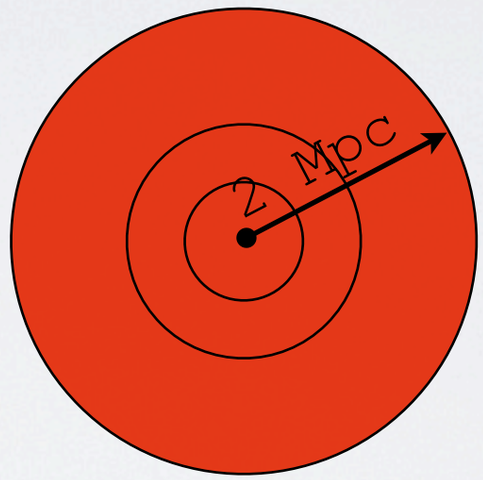
```
sp = pf.h.sphere(center,
                 (2, “mpc”))
```



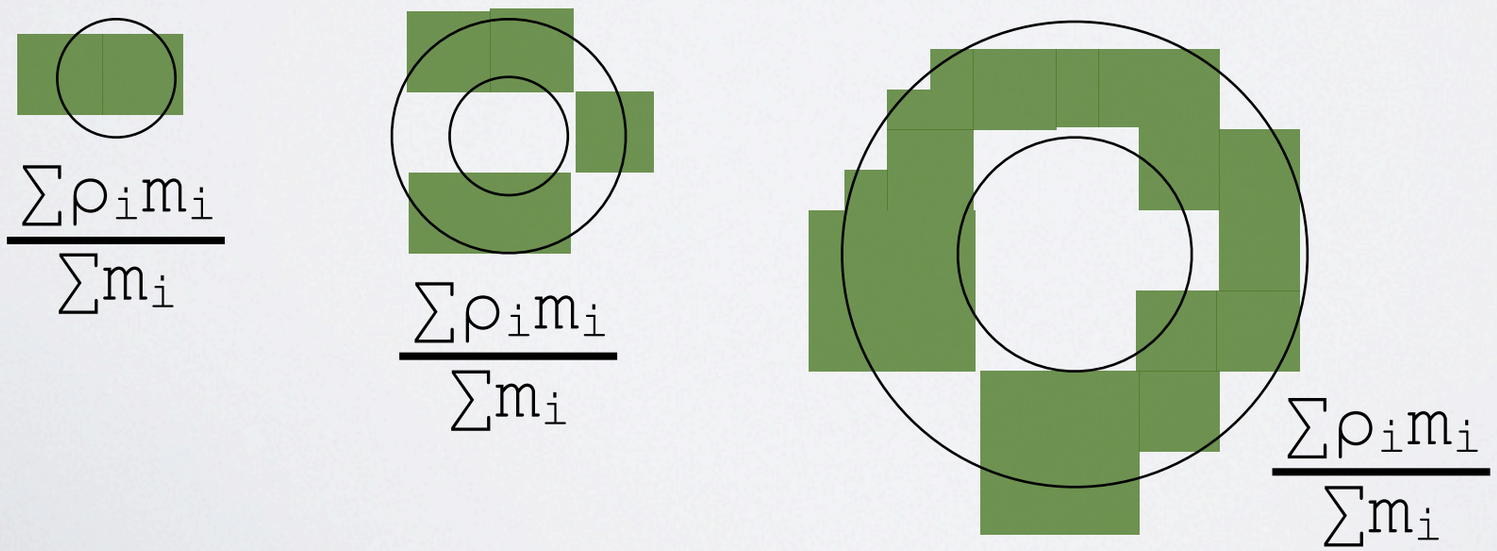

```
pc = PlotCollection(pf, center=center)
pc.add_phase_object(sp, [“RadiusMpc”, “Density”])
```

Profiles allow you to plot one field versus another.

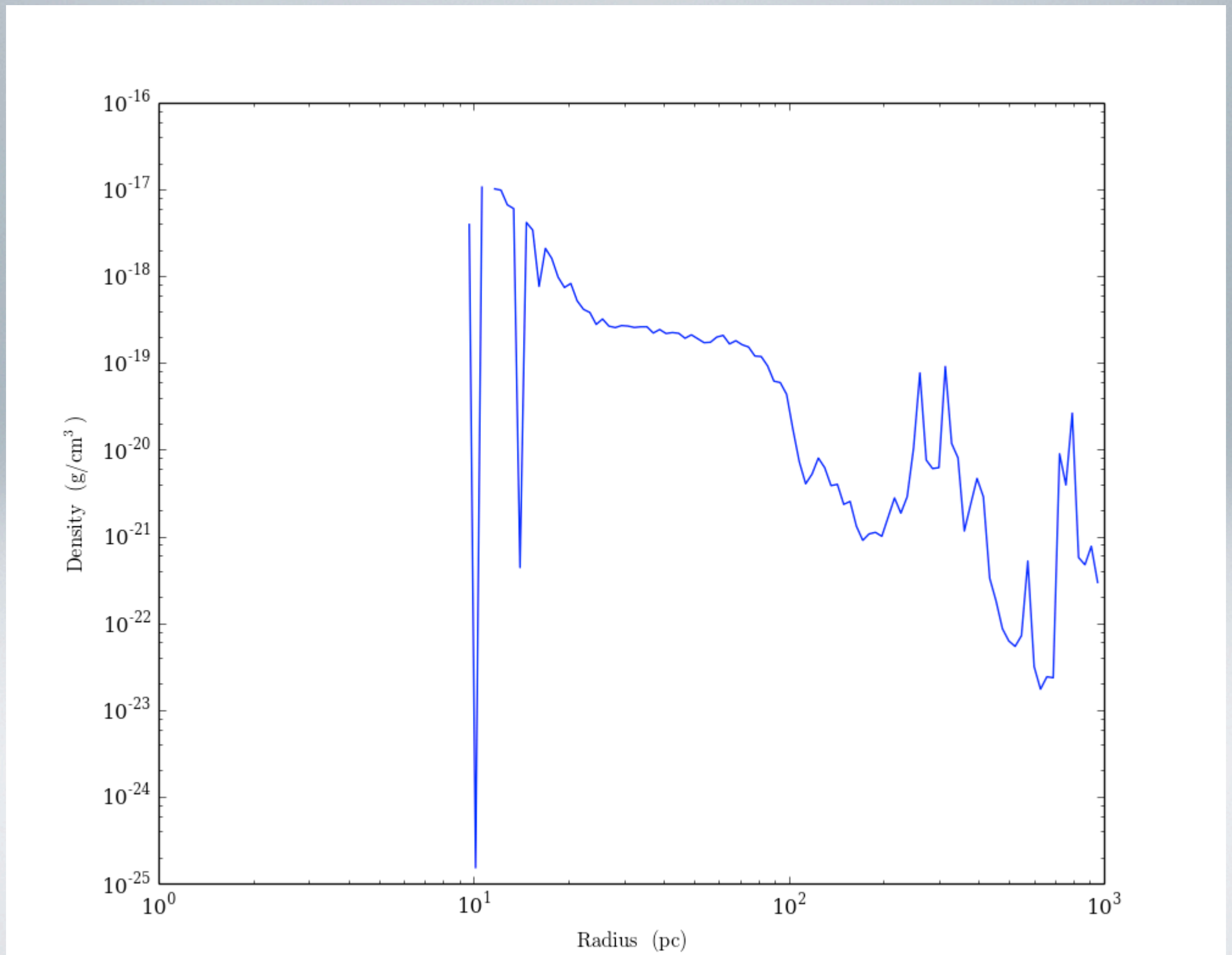
```
sp = pf.h.sphere(center,
                 (2, “mpc”))
```




```
pc = PlotCollection(pf, center=center)
pc.add_phase_object(sp, [“RadiusMpc”, “Density”])
```



Profiles - “08_profiles.py”

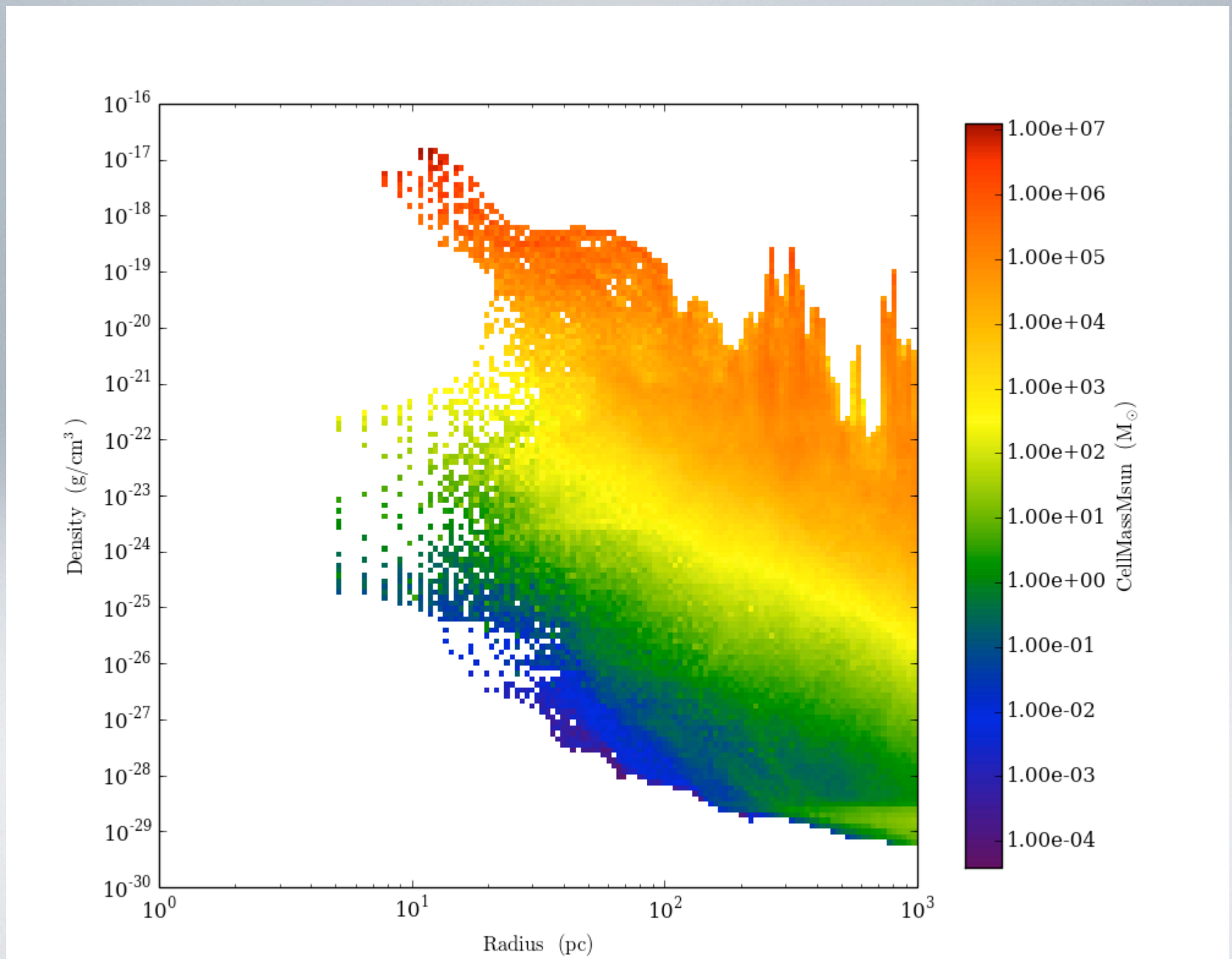


Challenge:

Make profiles of Density vs. Radiuspc for a disk object and compare with profiles of Density vs. "Height."

A phase plot is really just a 2-dimensional profile plot.

Phase plot - "09_phase_plot.py"

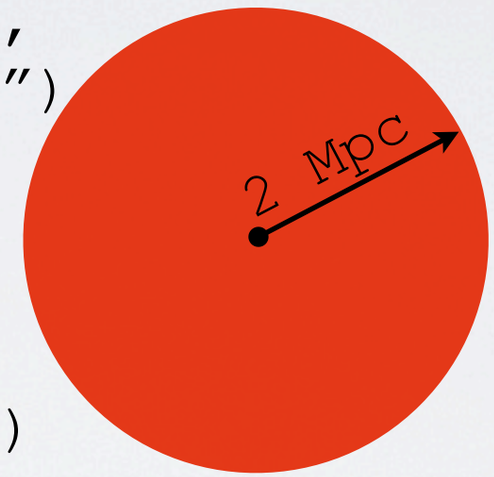


Challenge:

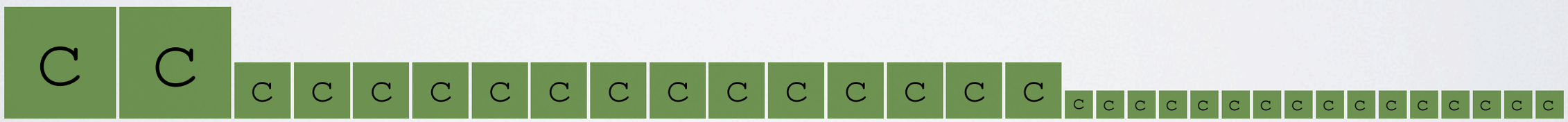
Make phase plots with "Height" instead of Radiuspc. Make phase plots of Radius vs. Height.

Creating new fields is easy.

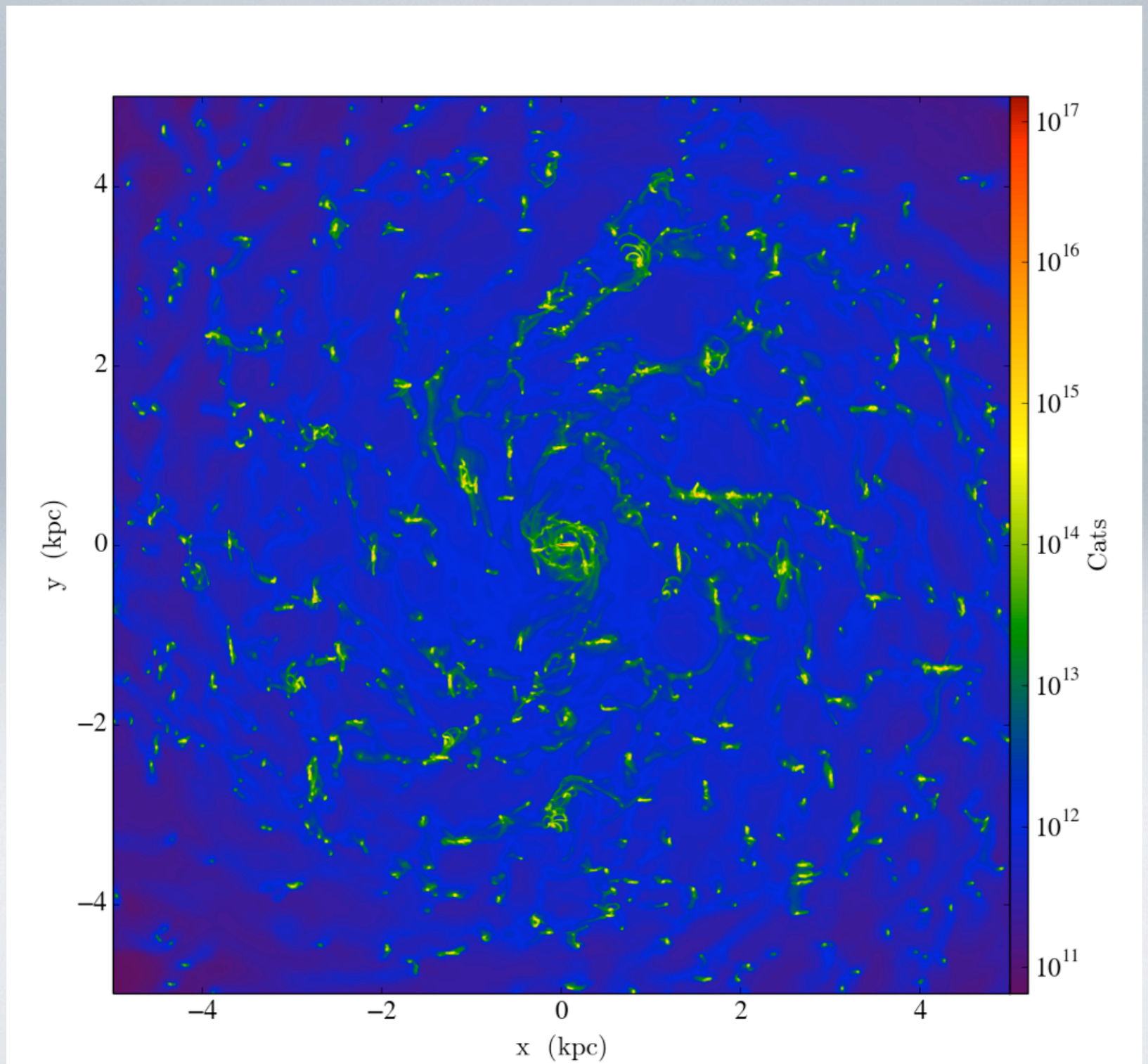
```
sp = pf.h.sphere(center,
                 (2, “mpc”))
@derived_field(name=”Cats”,
               units=”meow”)
def Cats(field, data):
    # this is a cat?
    return data[“x”] + \
      (data[“Density”] *
       data[“Temperature”])
```



sp[“Cats”]



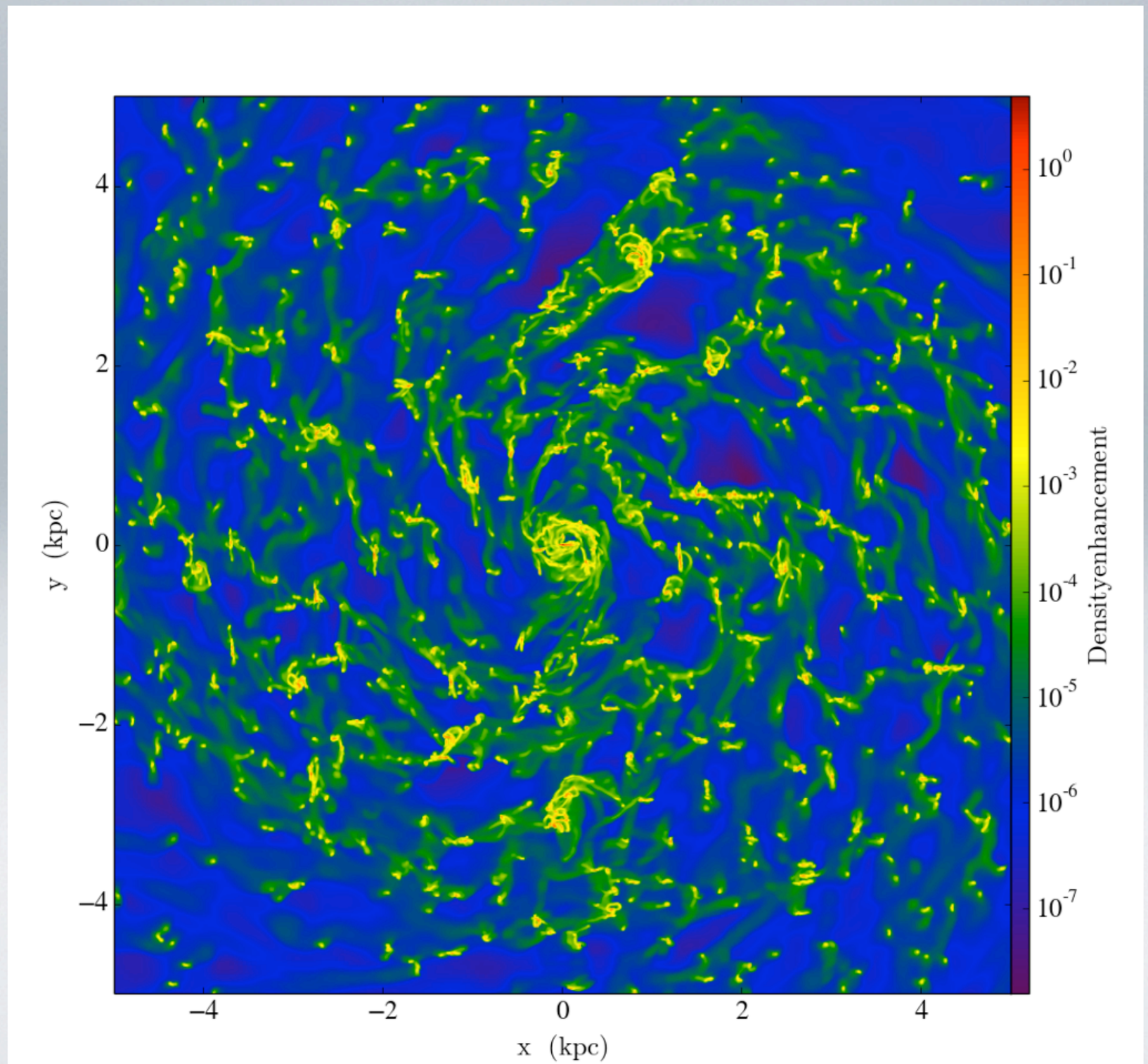
New fields - “10_new_fields.py”



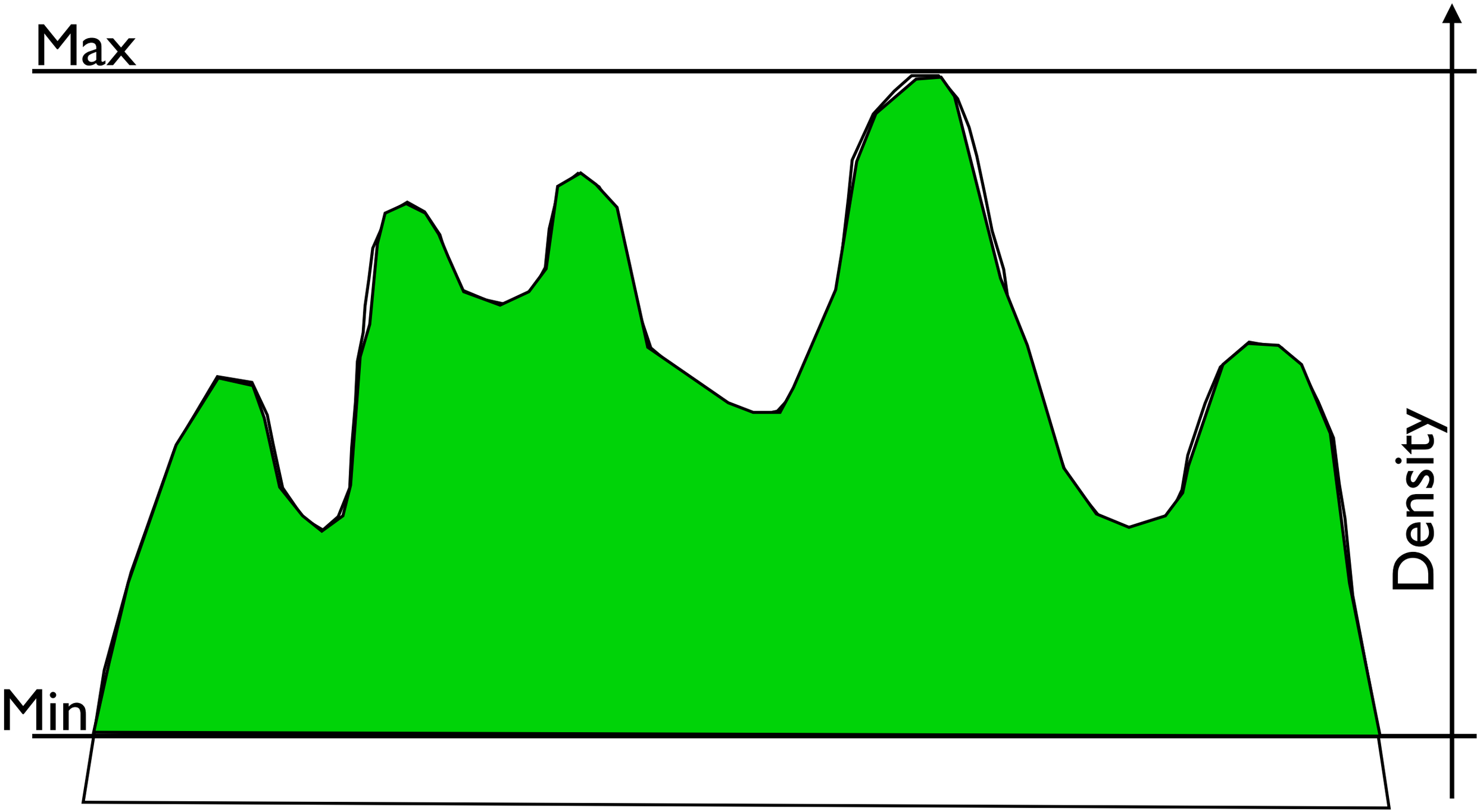
Advanced fields - “11_advanced_fields.py”

```
from yt.mods import *

@derived_field(name="DensityEnhancement")
def DensityEnhancement(field, data):
    my_density = data.get_field_parameter("mean_density")
    return data["Density"] / mean_density


pf = load("DD0200/R7_YC_0200")

my_sphere = pf.h.sphere(pf.domain_center, (10, "kpc"))
mean_density = my_sphere.quantities["WeightedAverageQuantity"]("Density",
                                                               "CellMassMsun")
my_sphere.set_field_parameter("mean_density", mean_density)
print "Mean density:", mean_density

plot = ProjectionPlot(pf, "z", "DensityEnhancement", weight_field="Density",
                      data_source=my_sphere, width=(10, "kpc"))
plot.save()
```

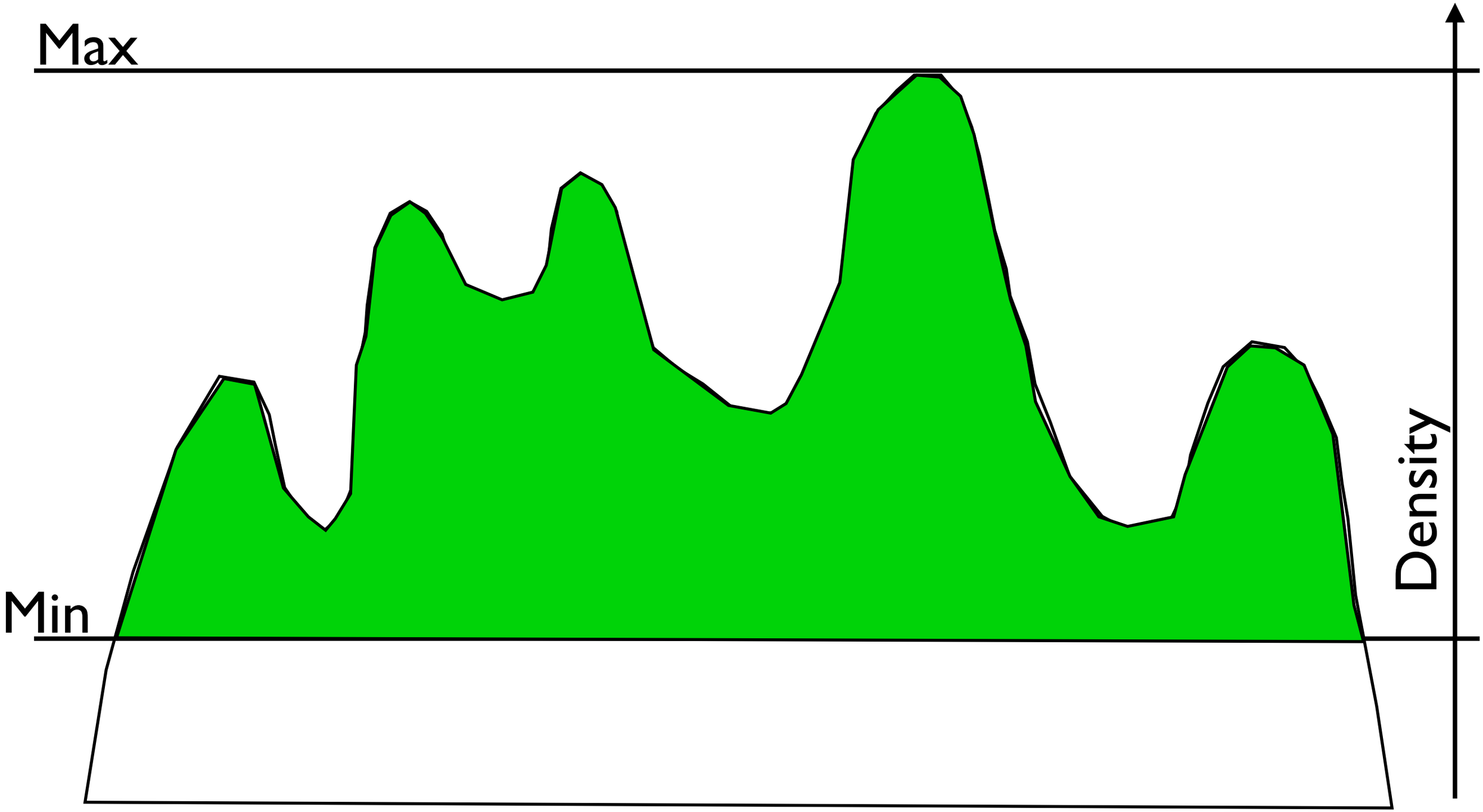
Advanced fields - “1 1_advanced_fields.py”



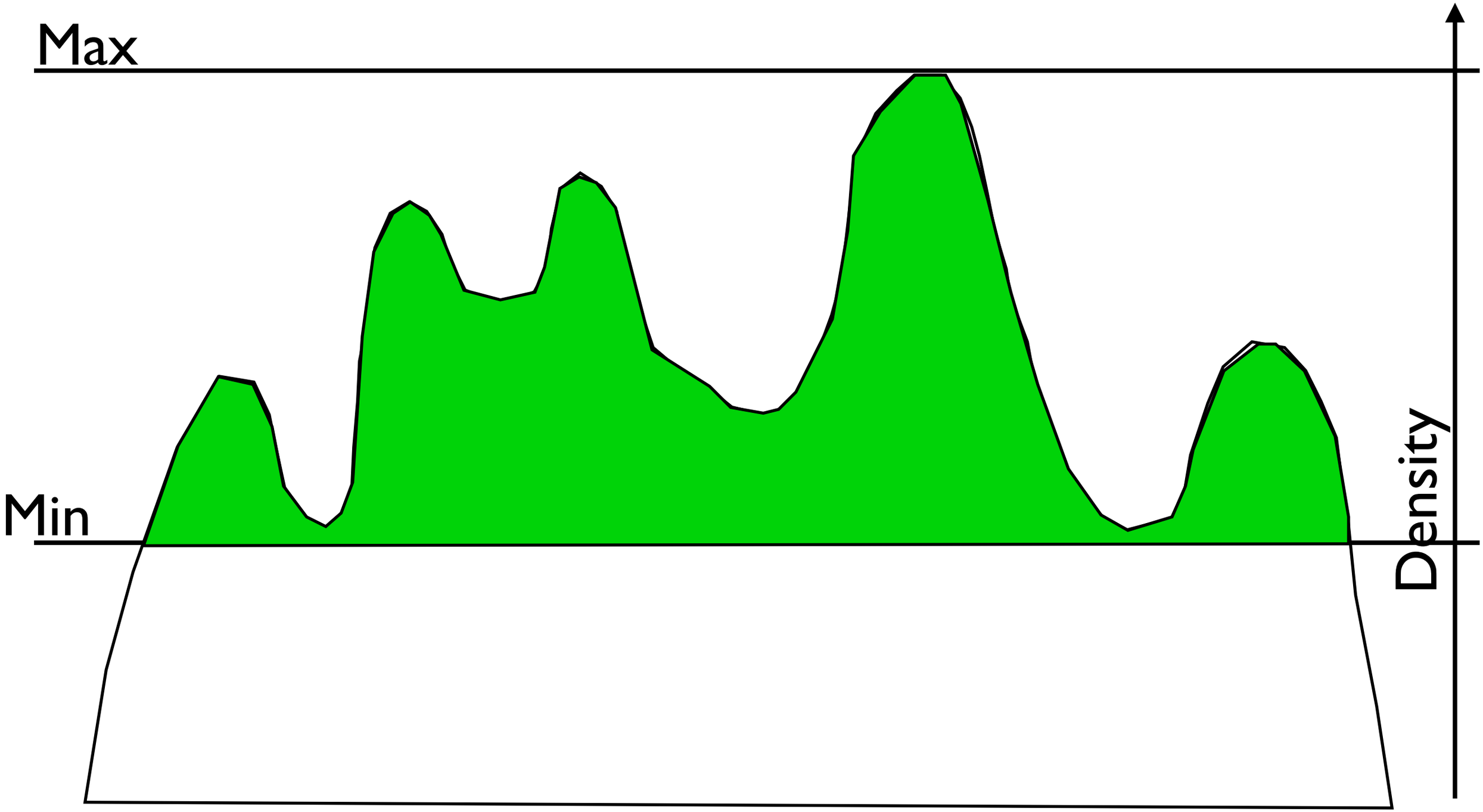
Finding Clumps



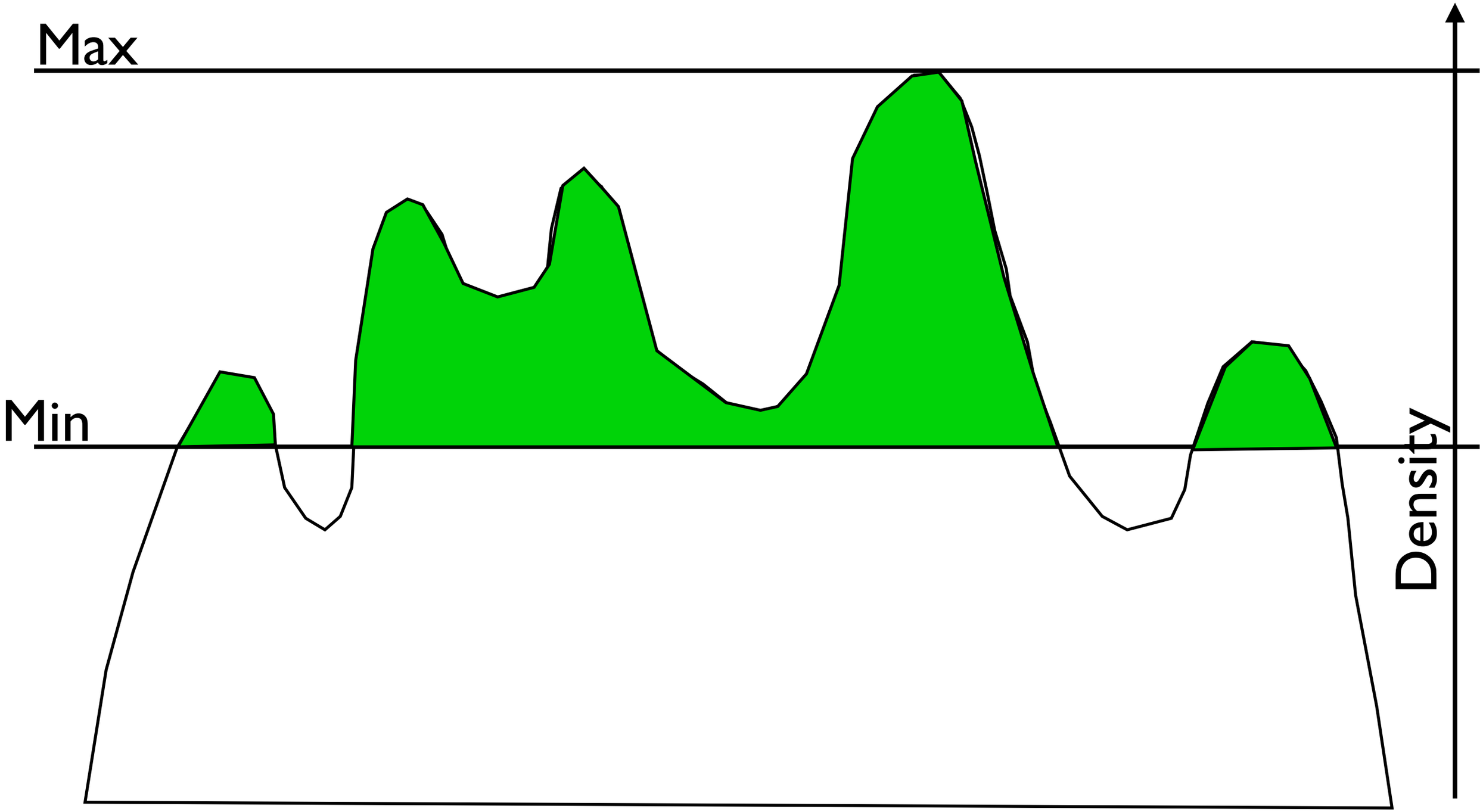
Finding Clumps



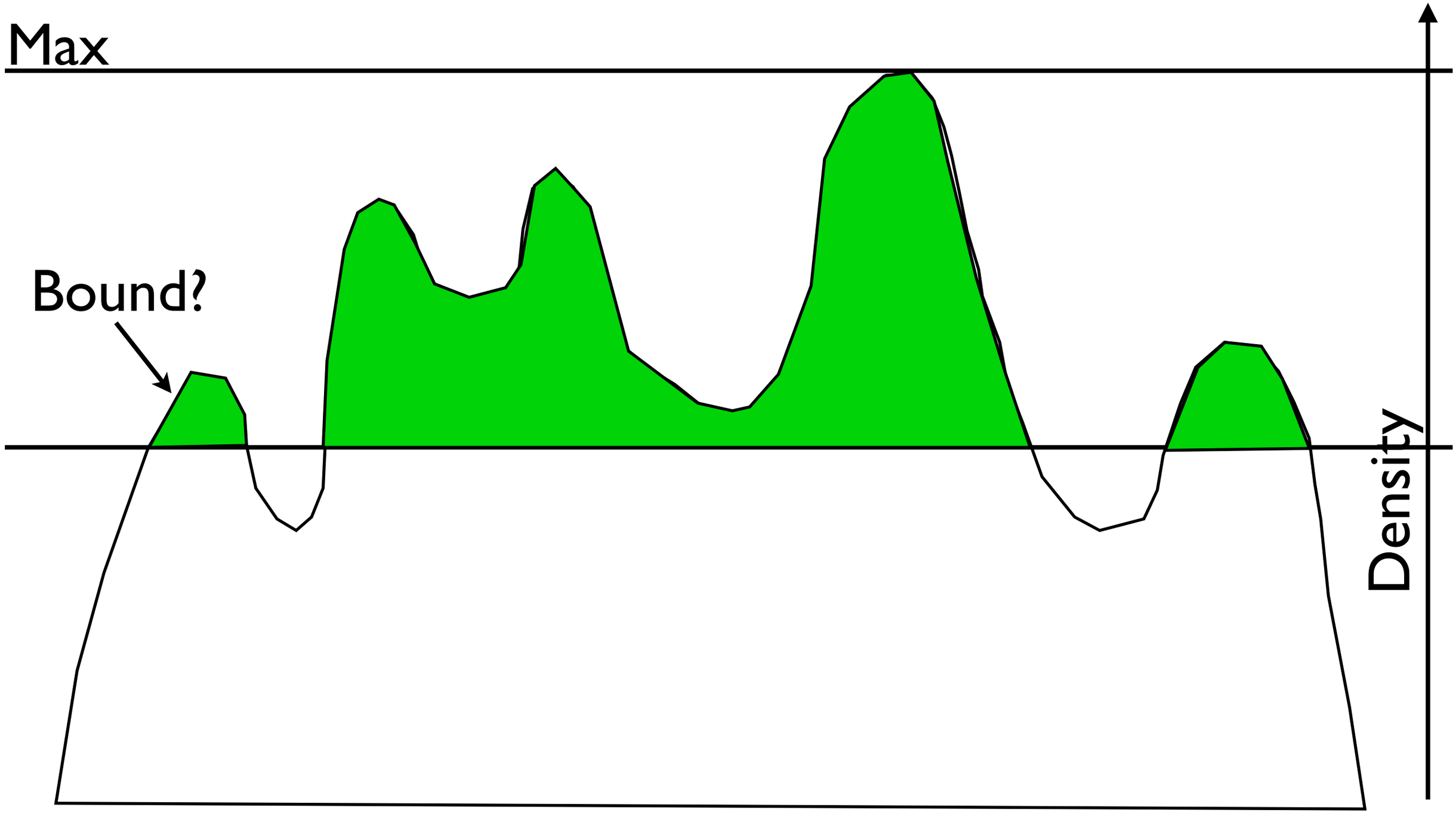
Finding Clumps



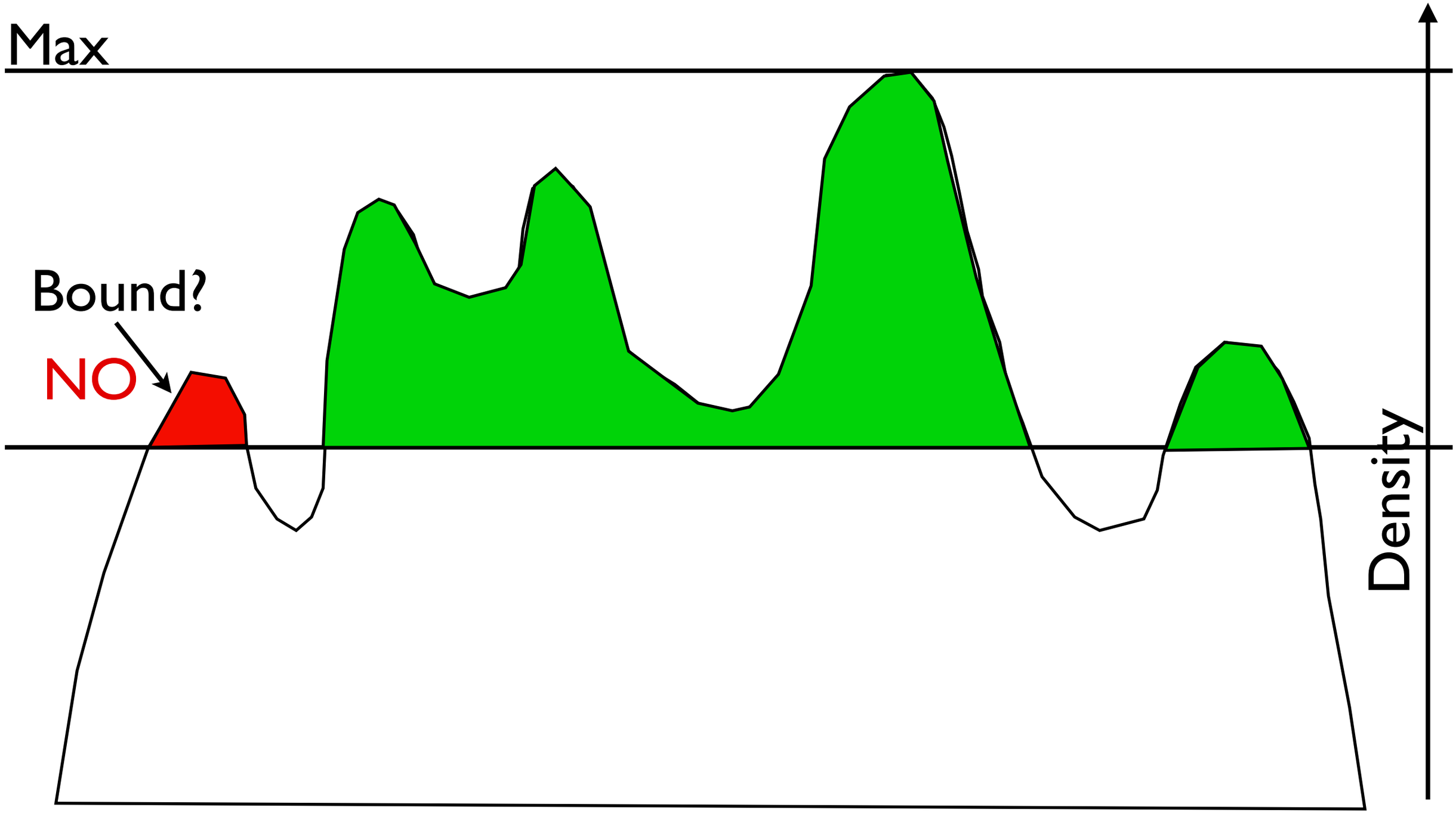
Finding Clumps



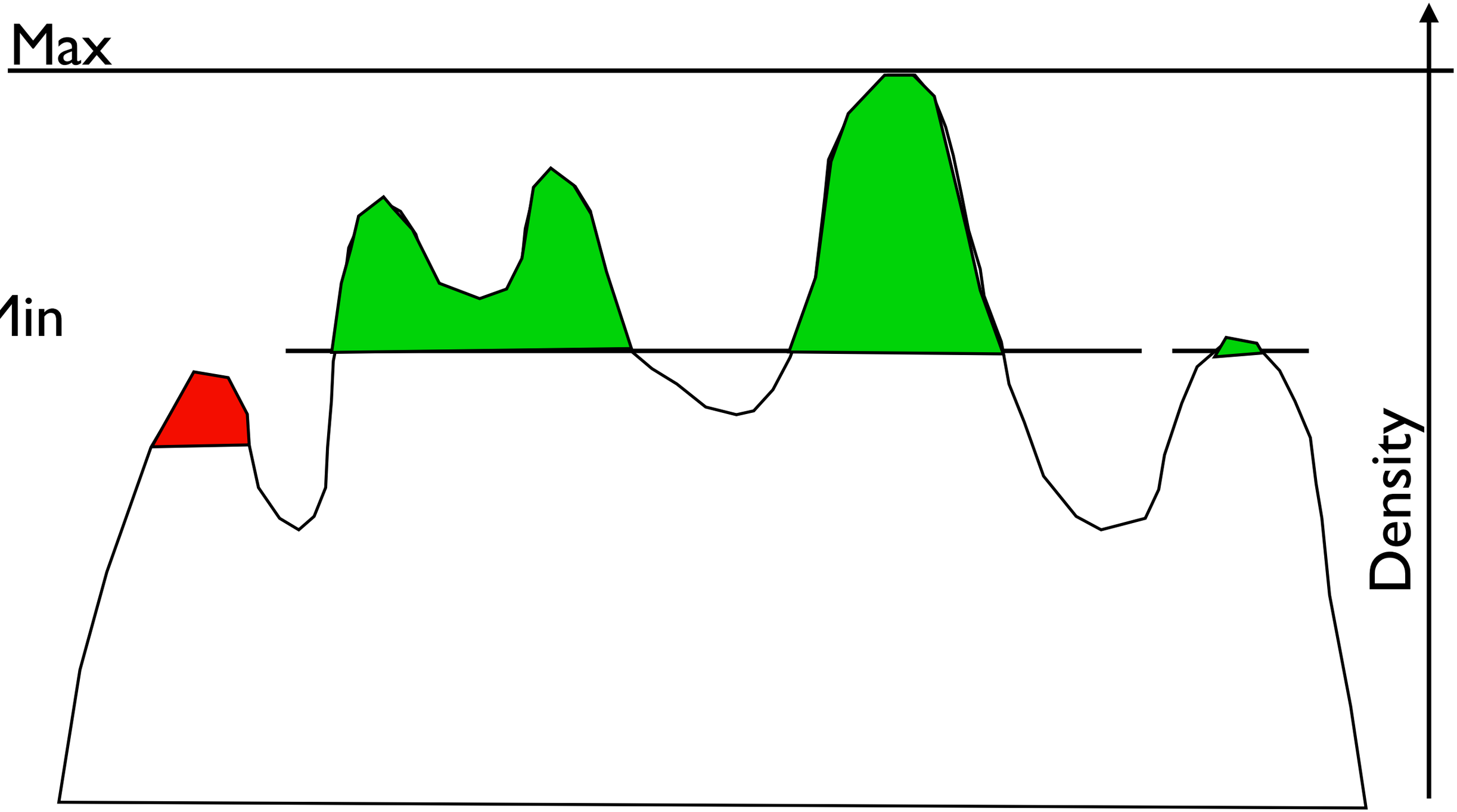
Finding Clumps



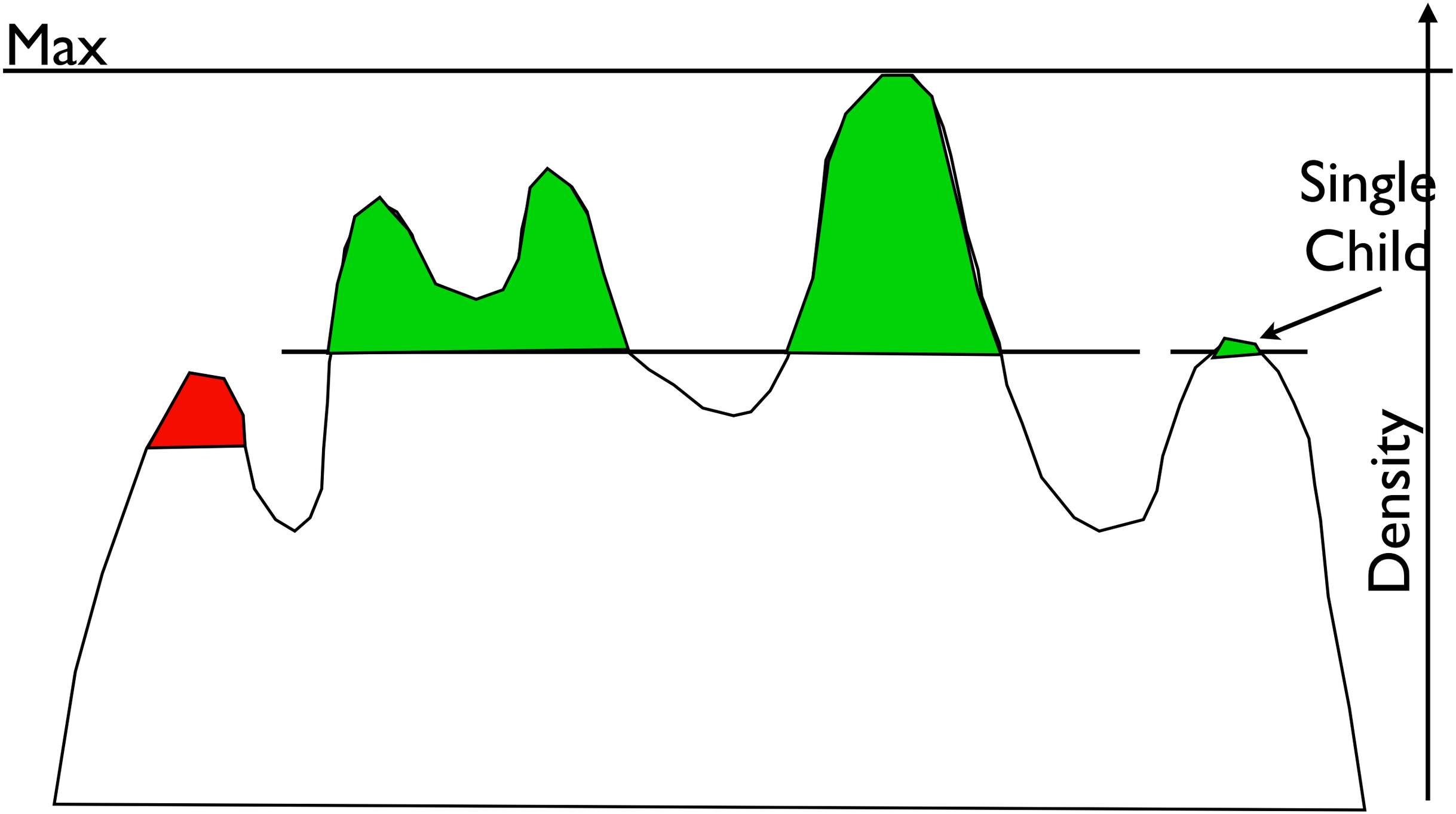
Finding Clumps



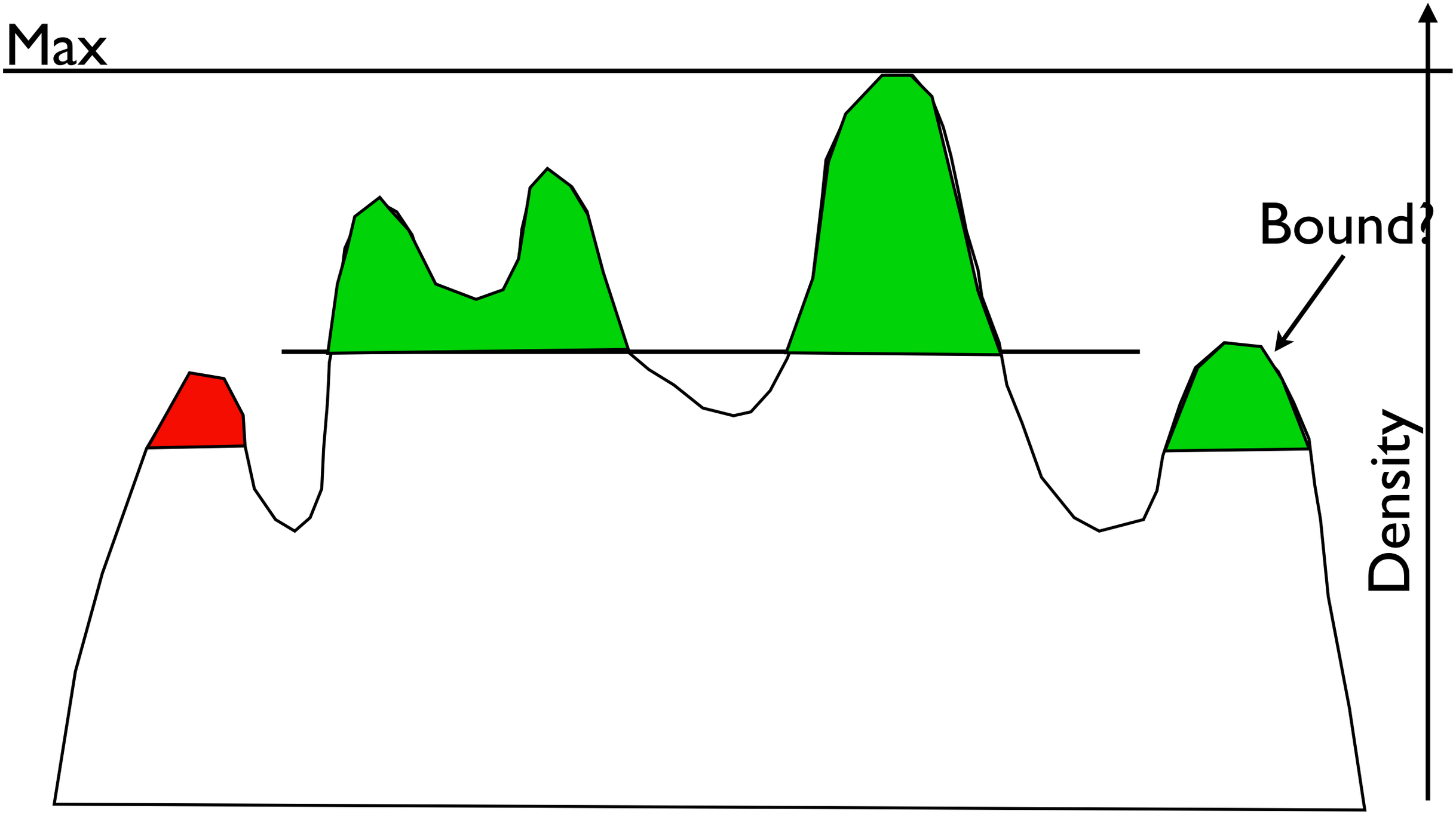
Finding Clumps



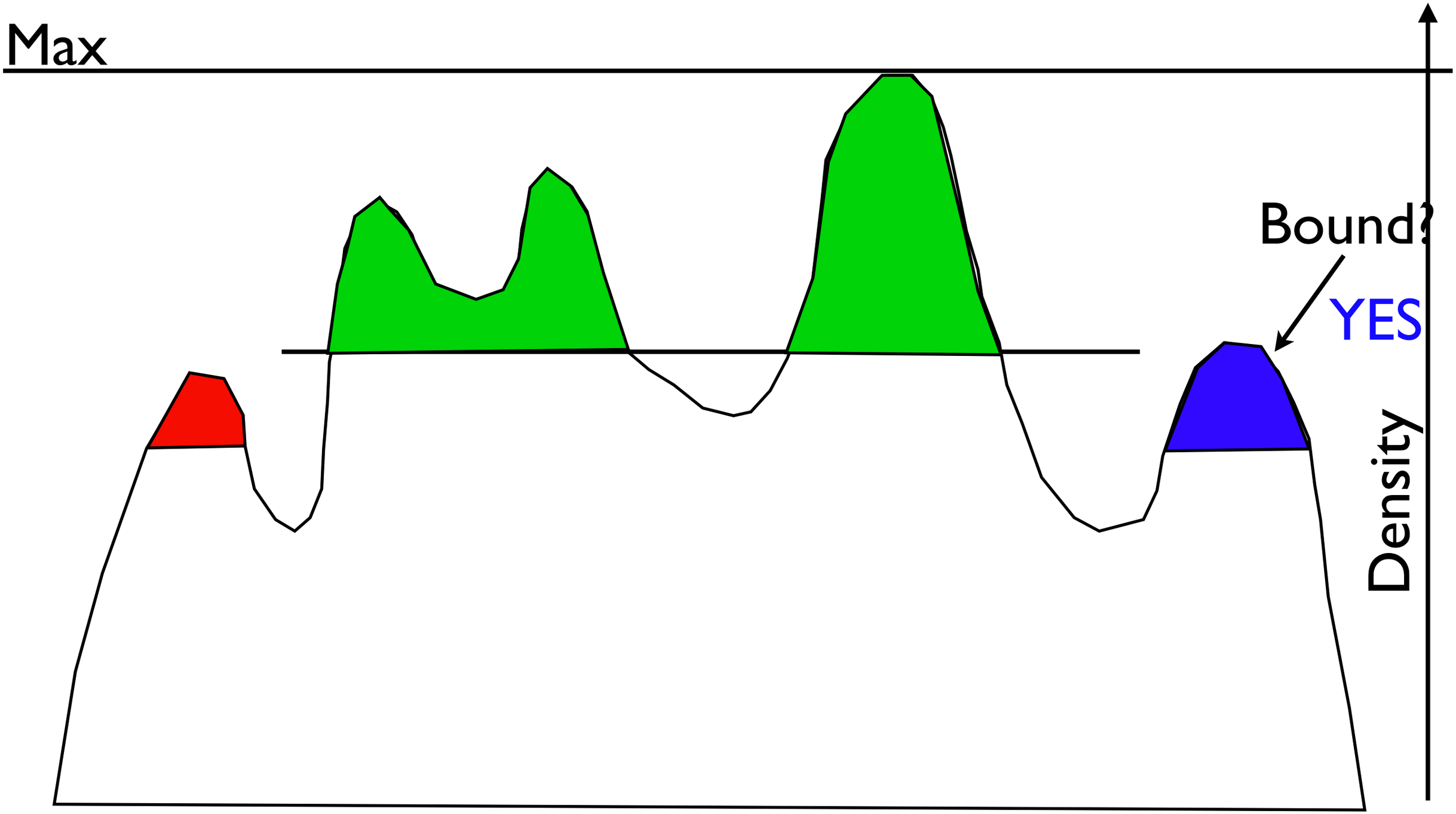
Finding Clumps



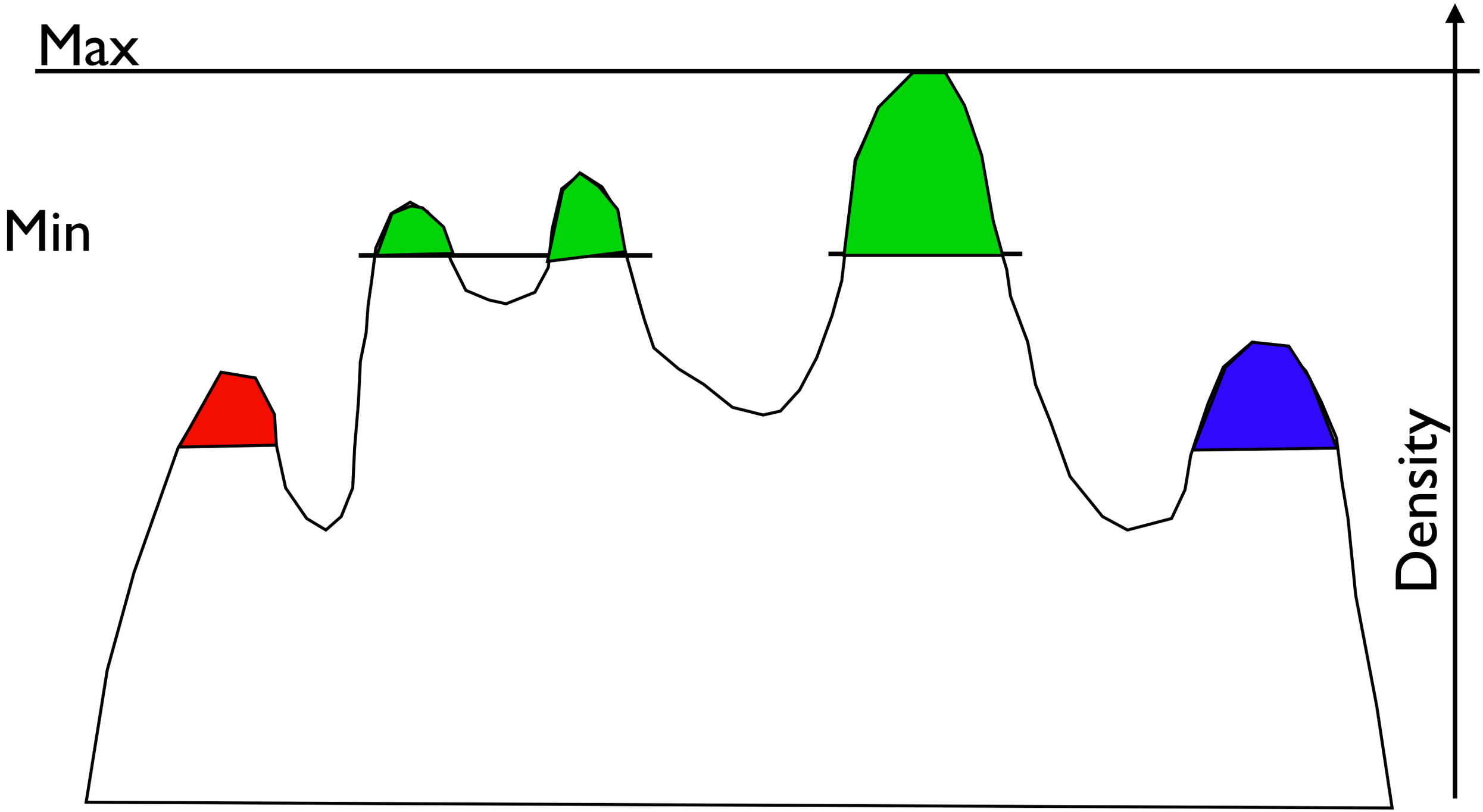
Finding Clumps



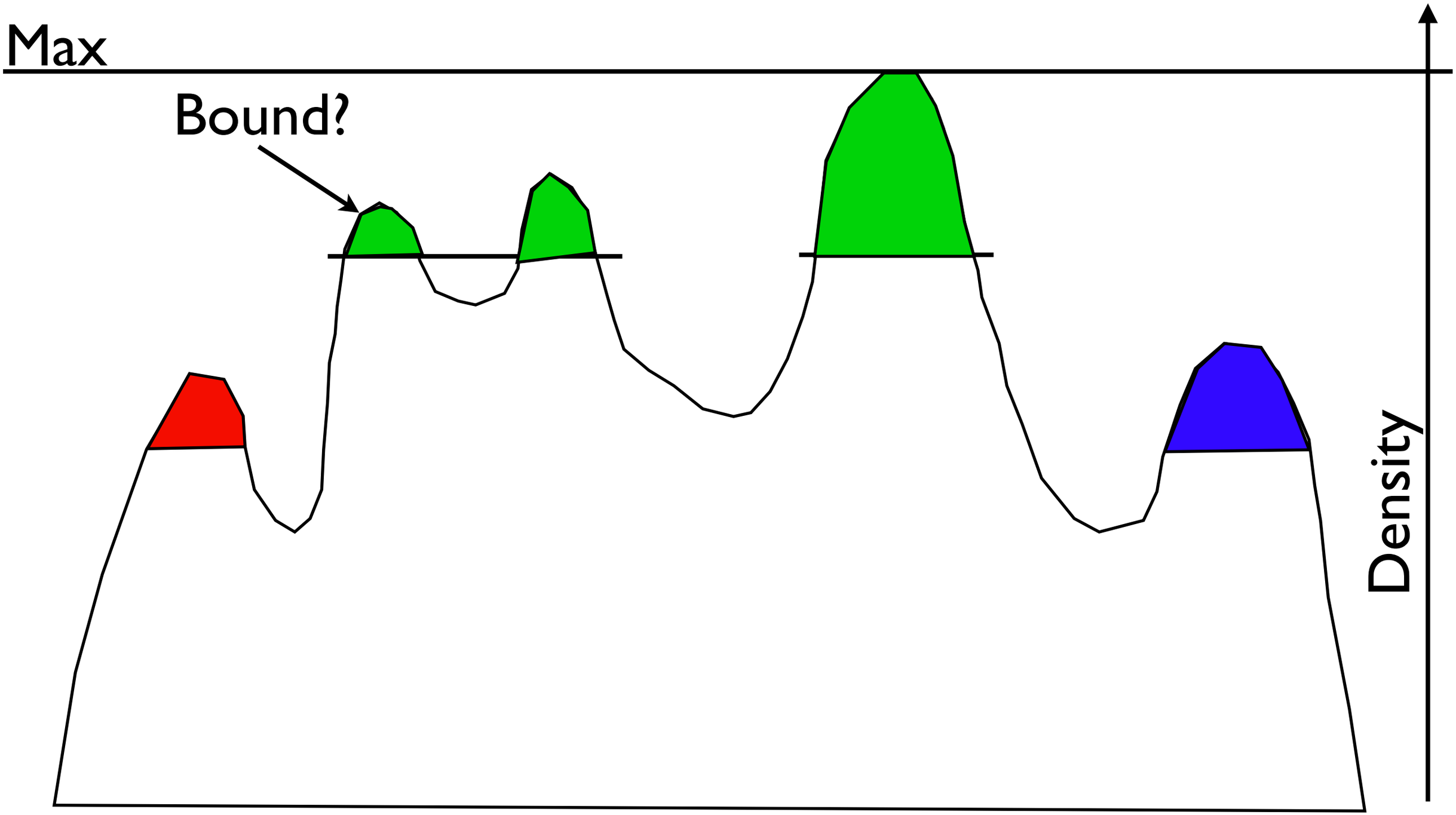
Finding Clumps



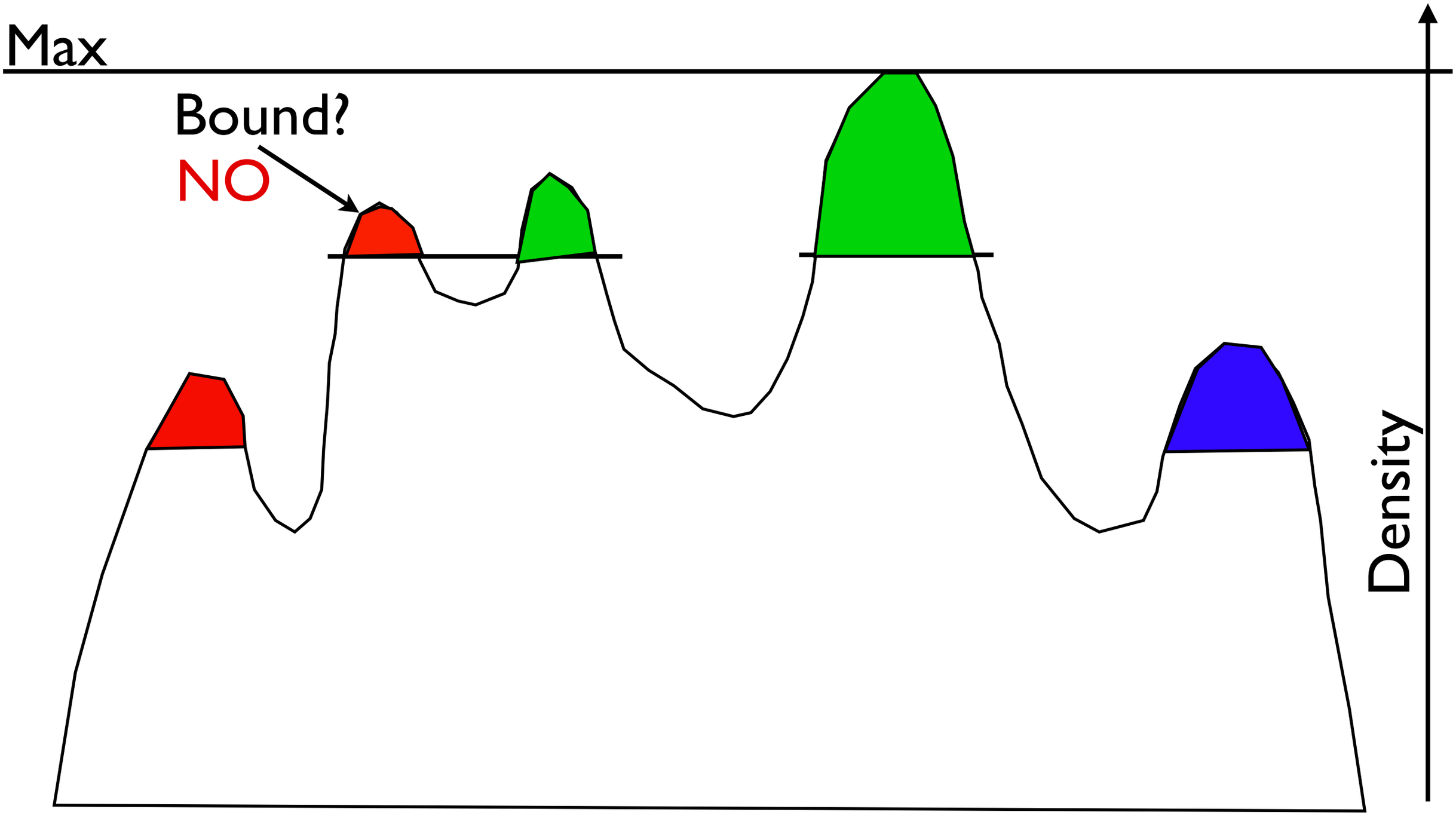
Finding Clumps



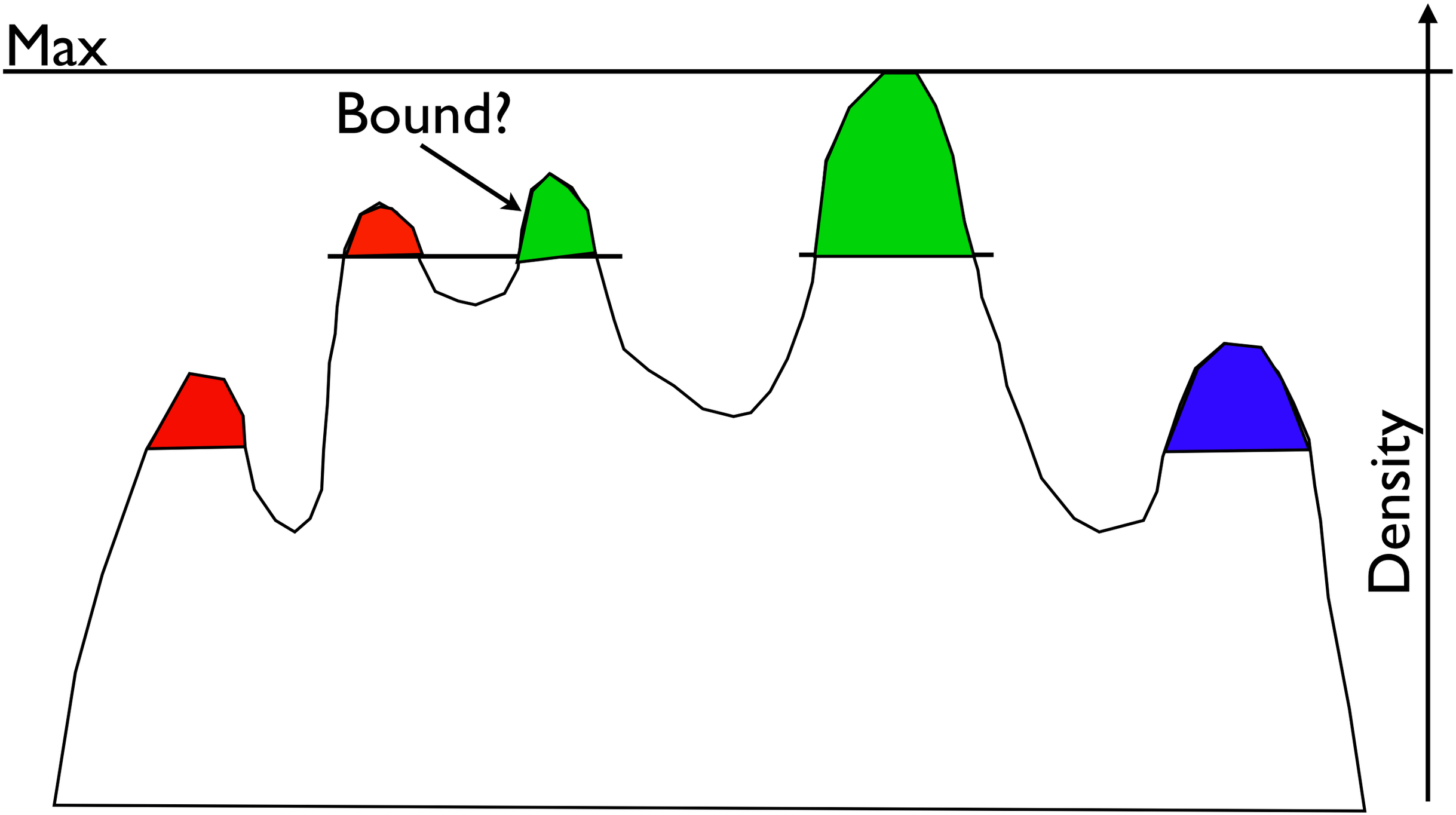
Finding Clumps



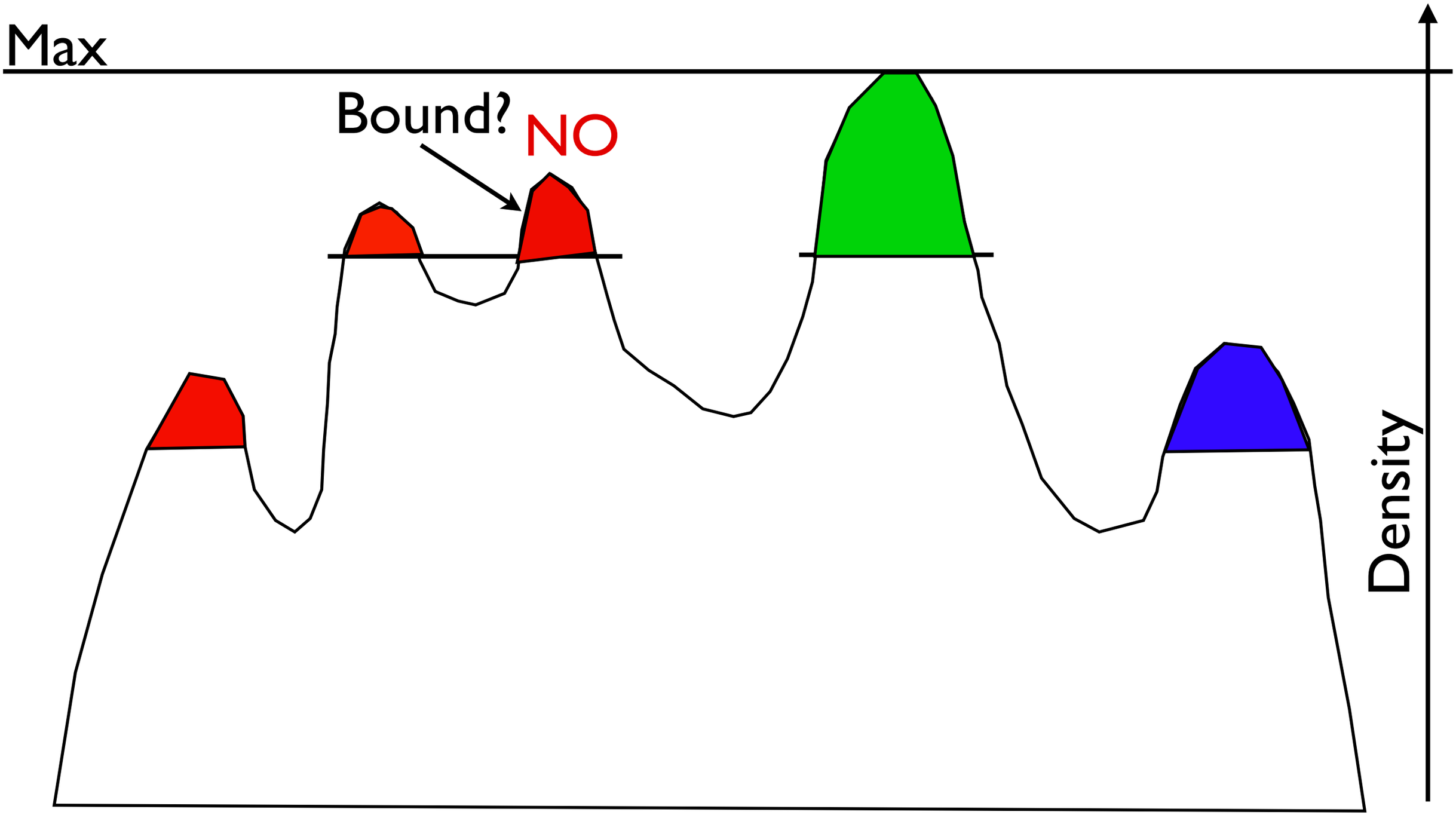
Finding Clumps



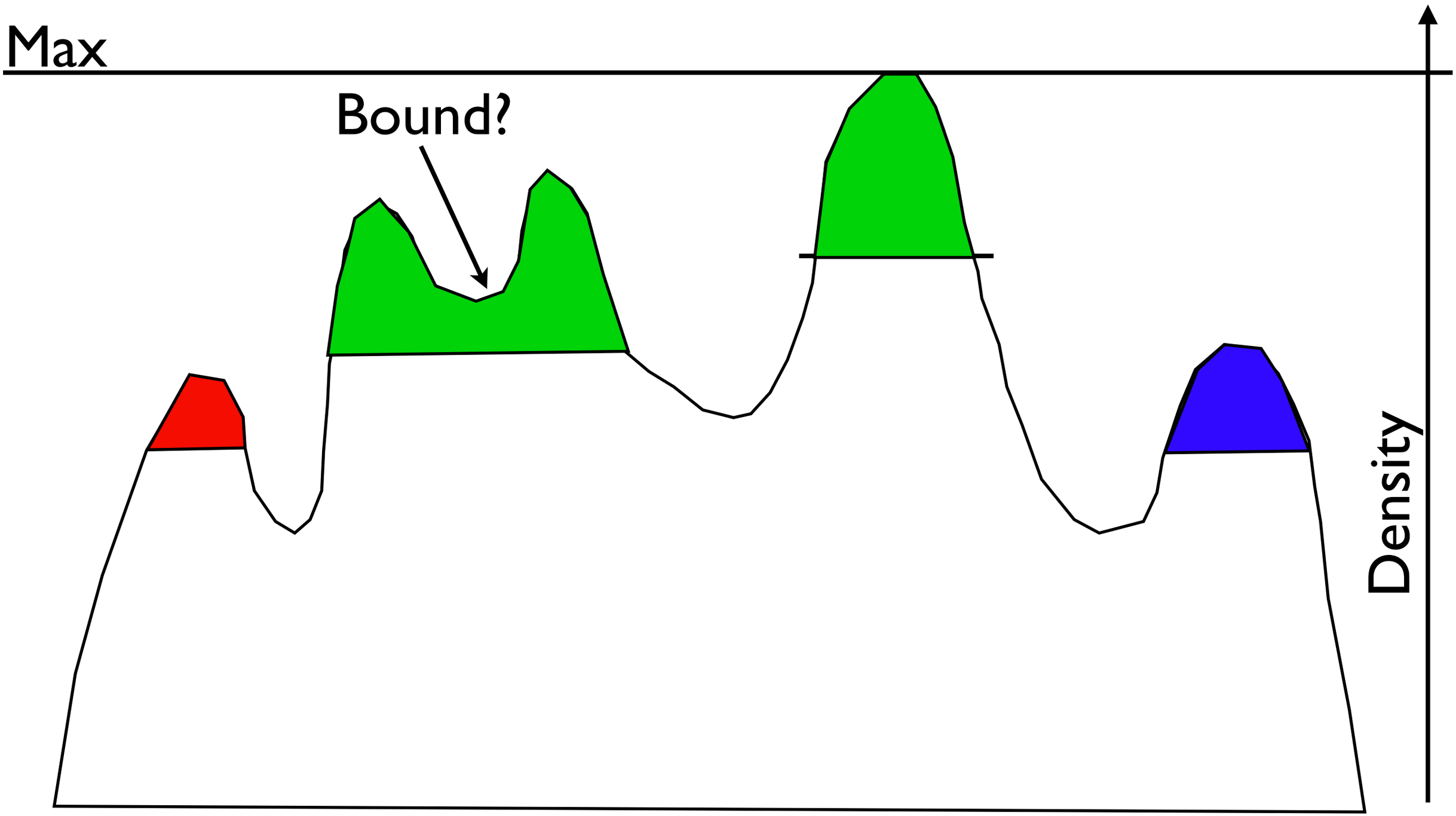
Finding Clumps



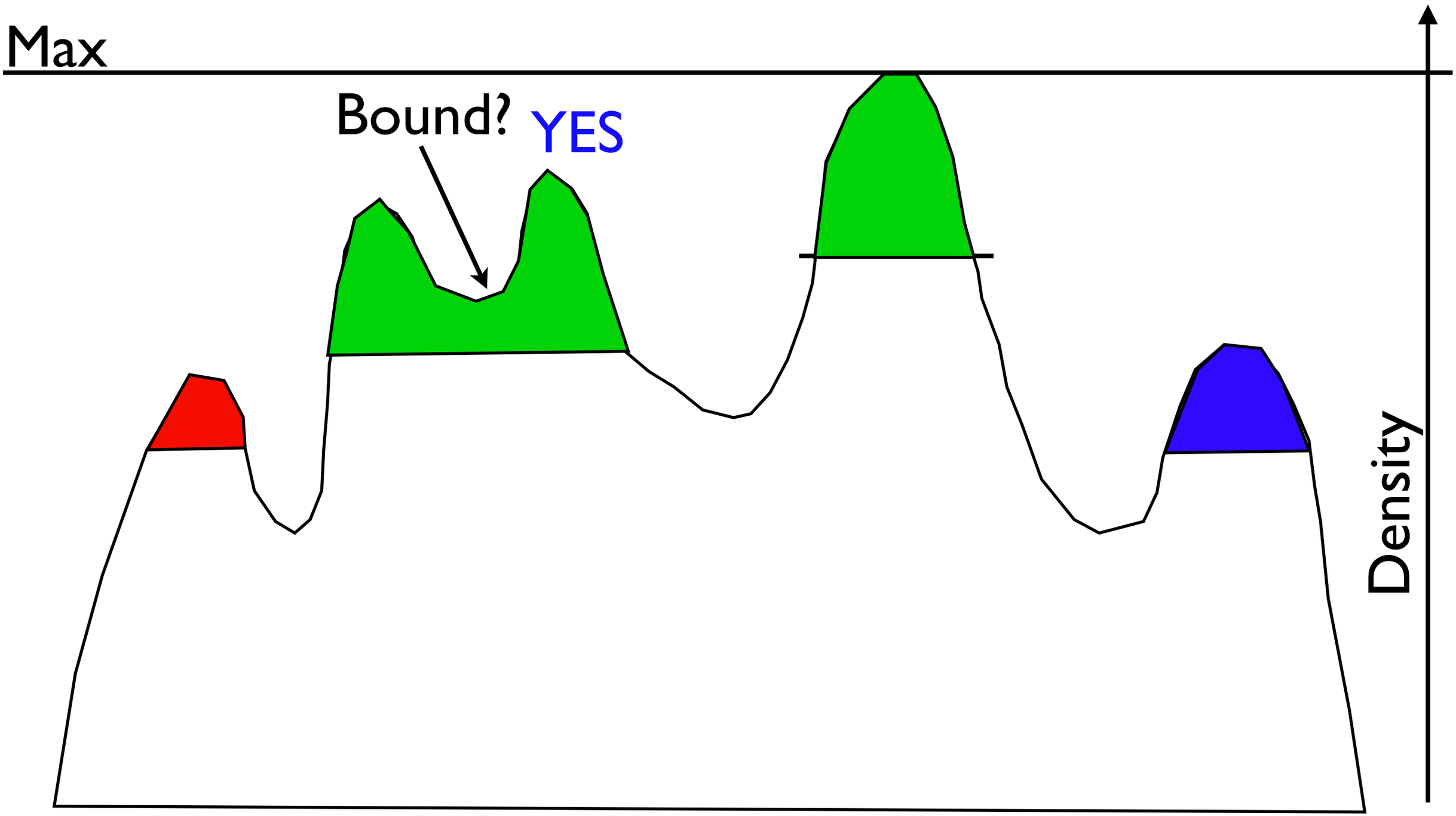
Finding Clumps



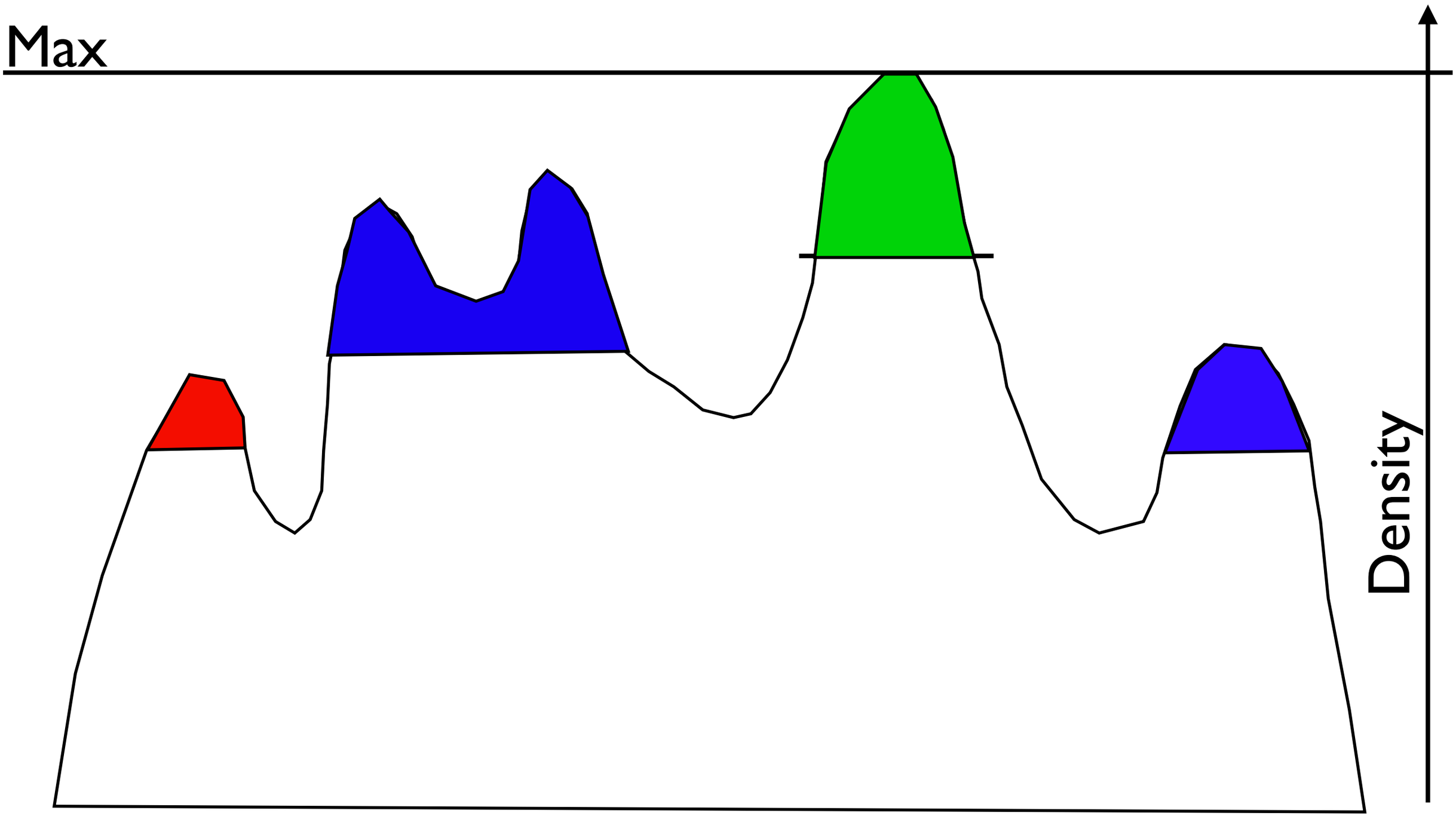
Finding Clumps



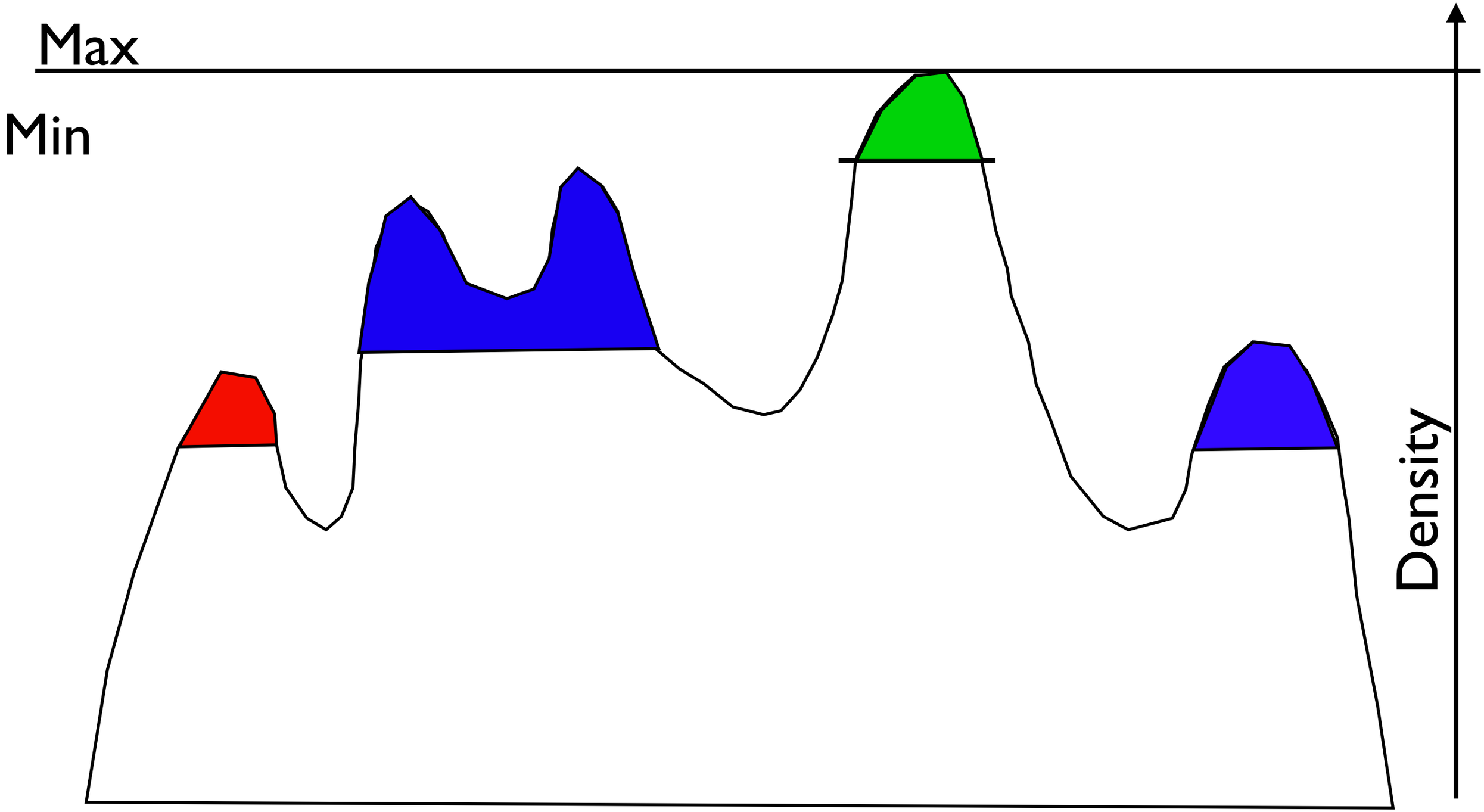
Finding Clumps



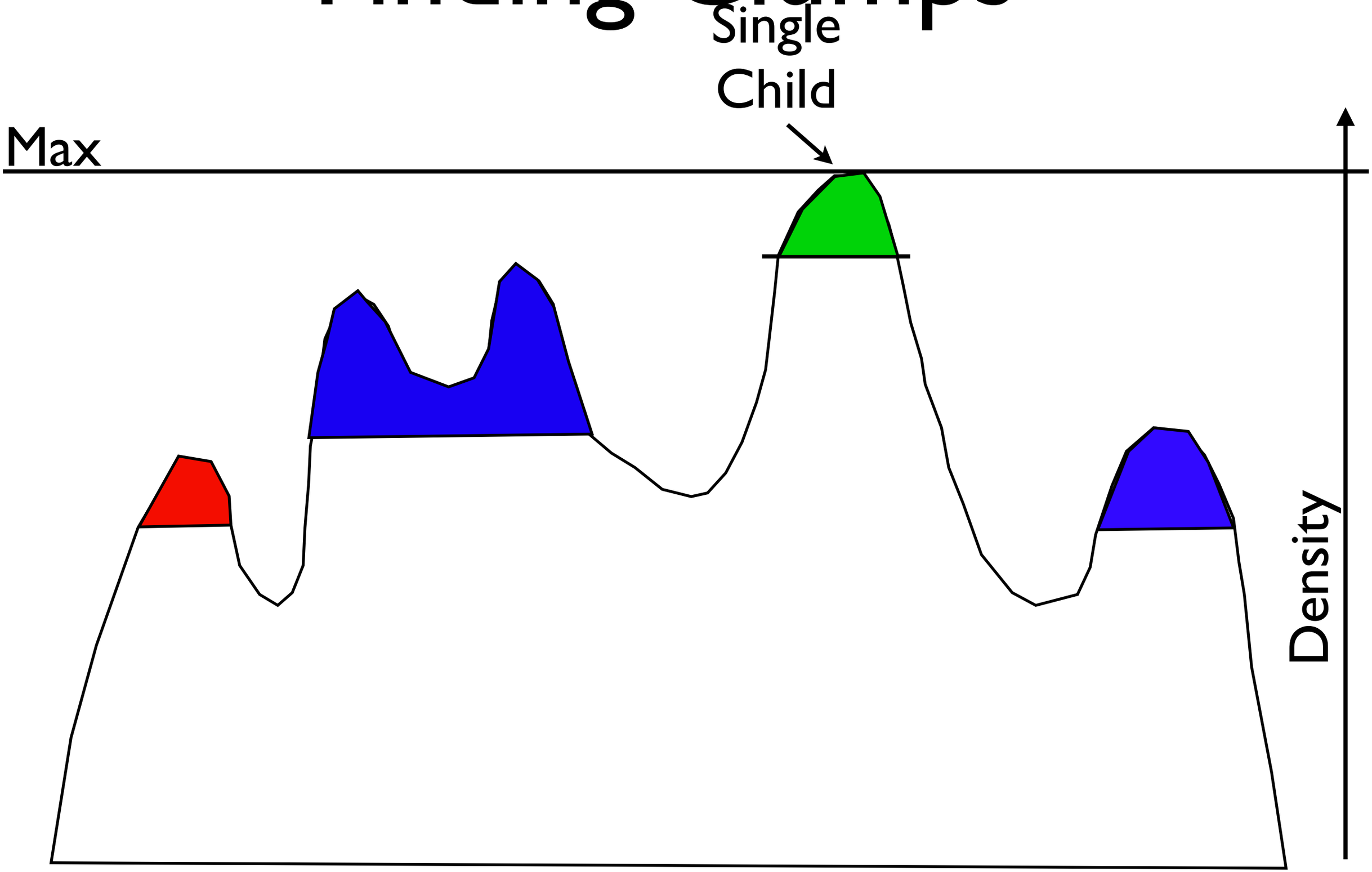
Finding Clumps



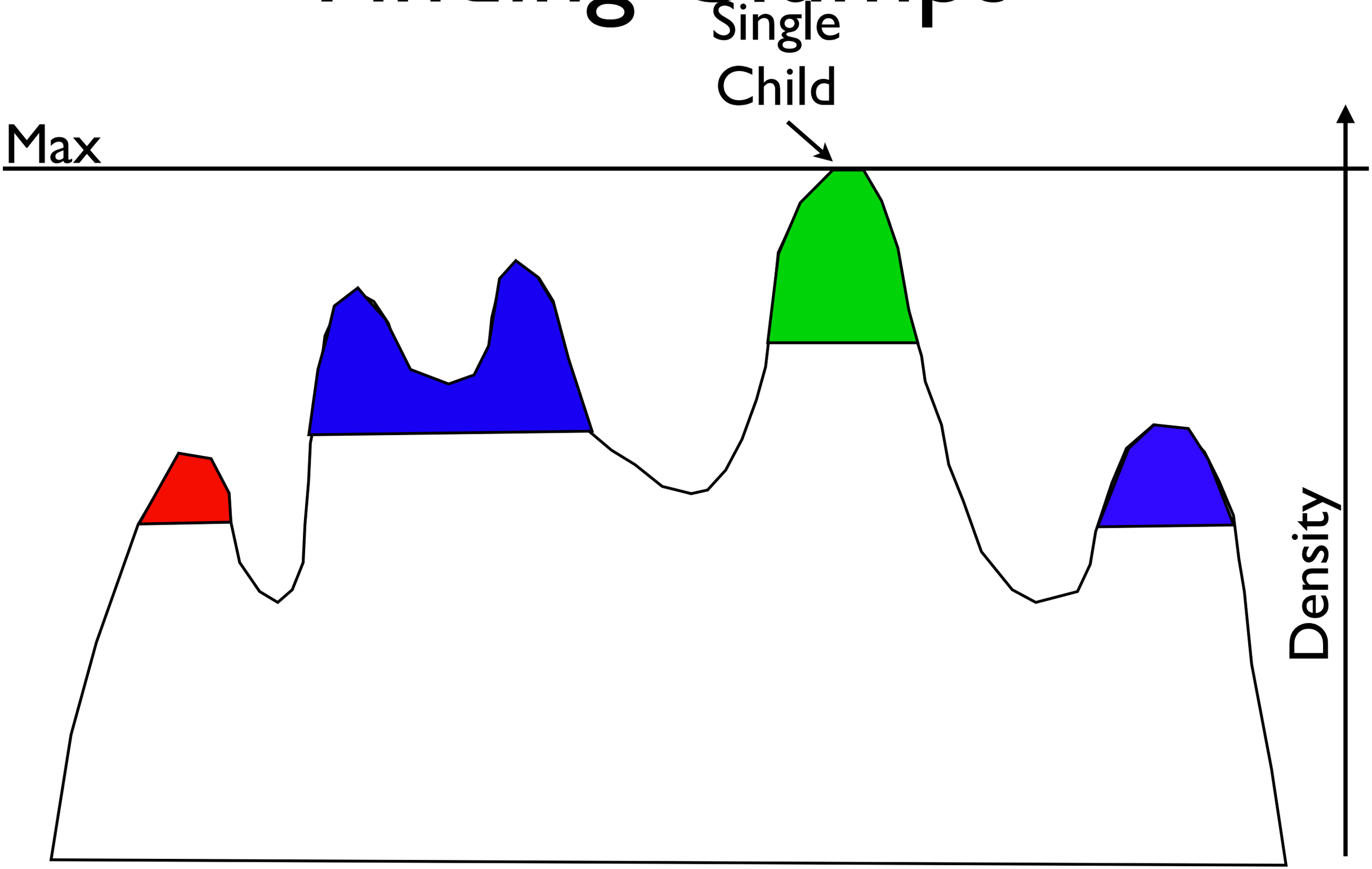
Finding Clumps



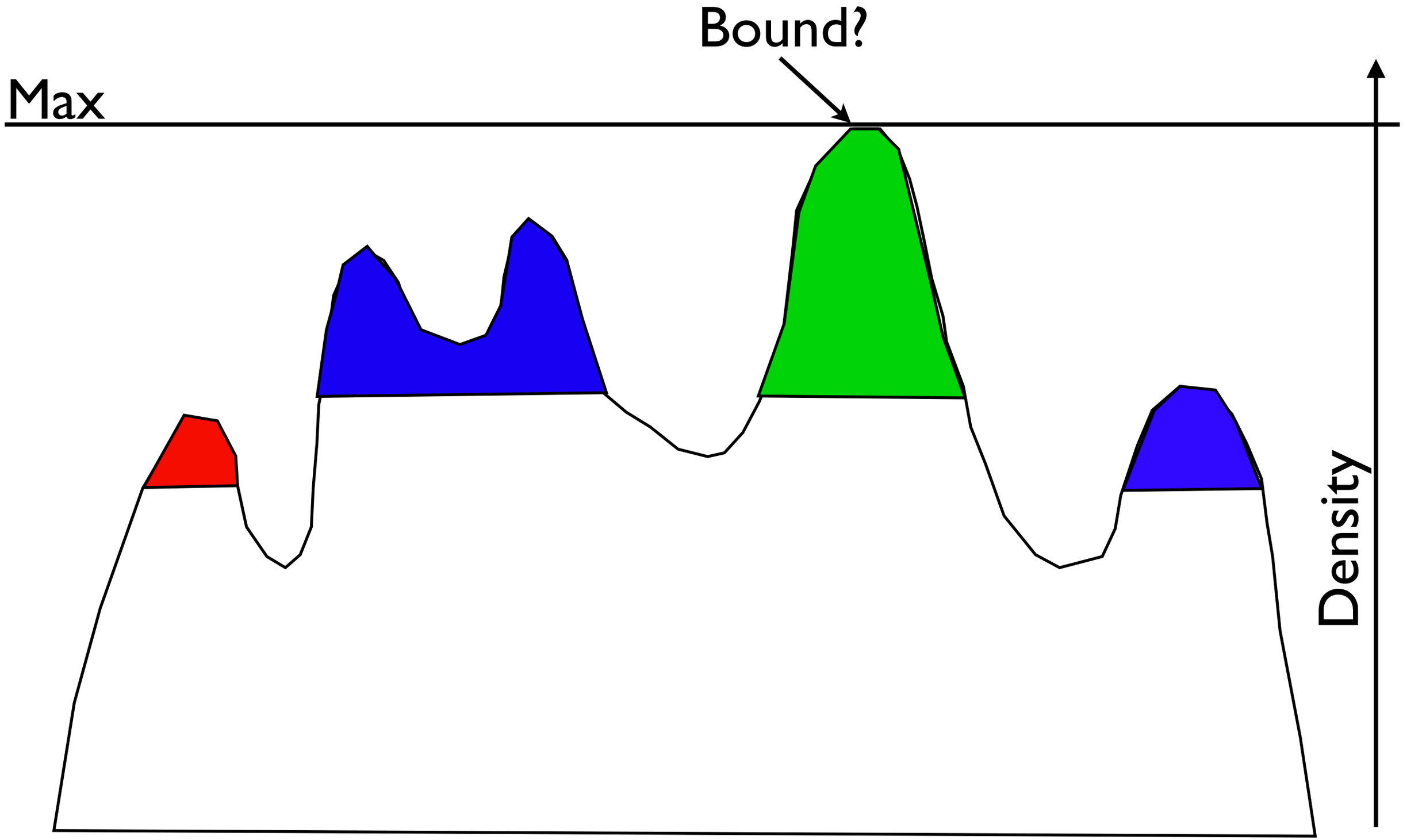
Finding Clumps



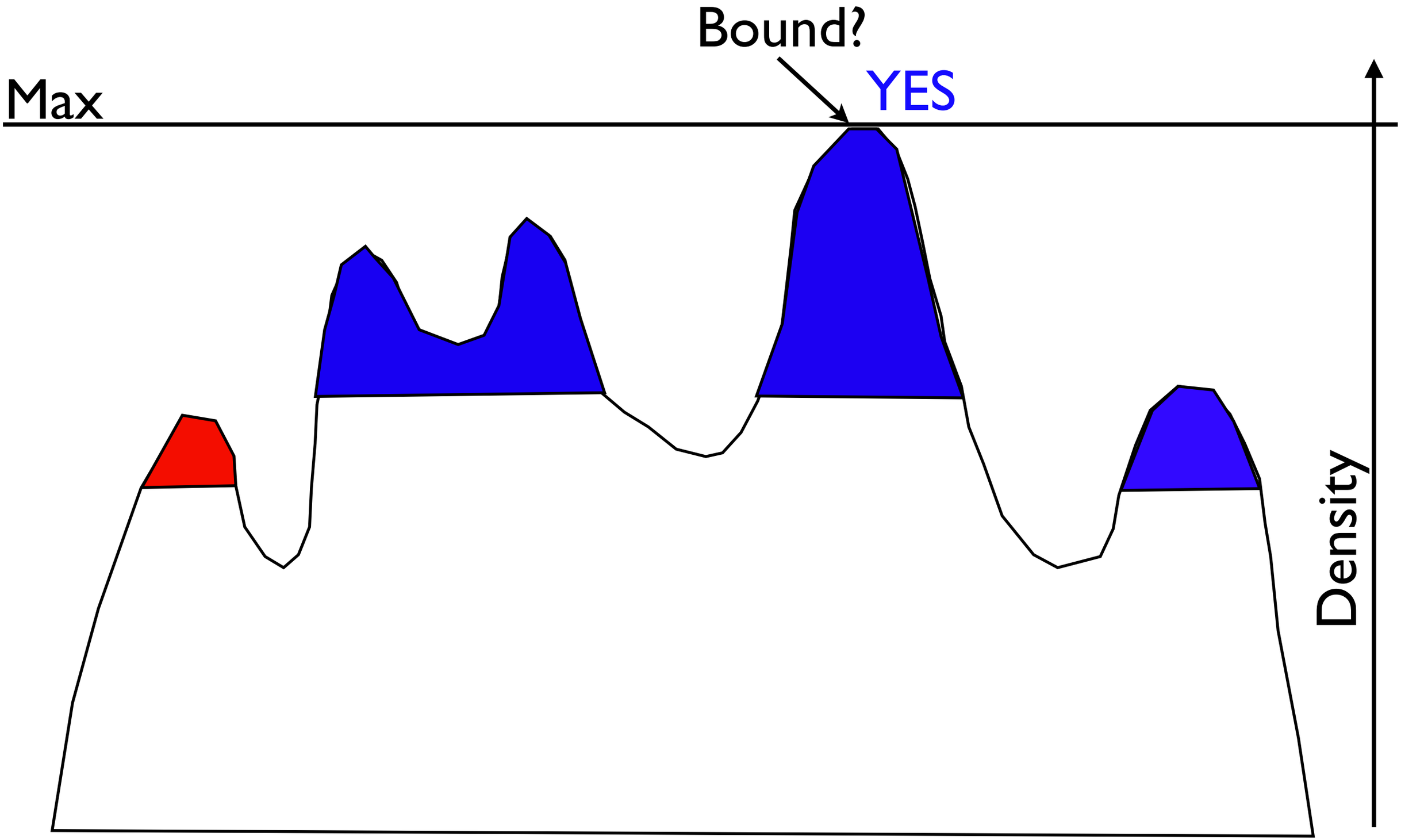
Finding Clumps



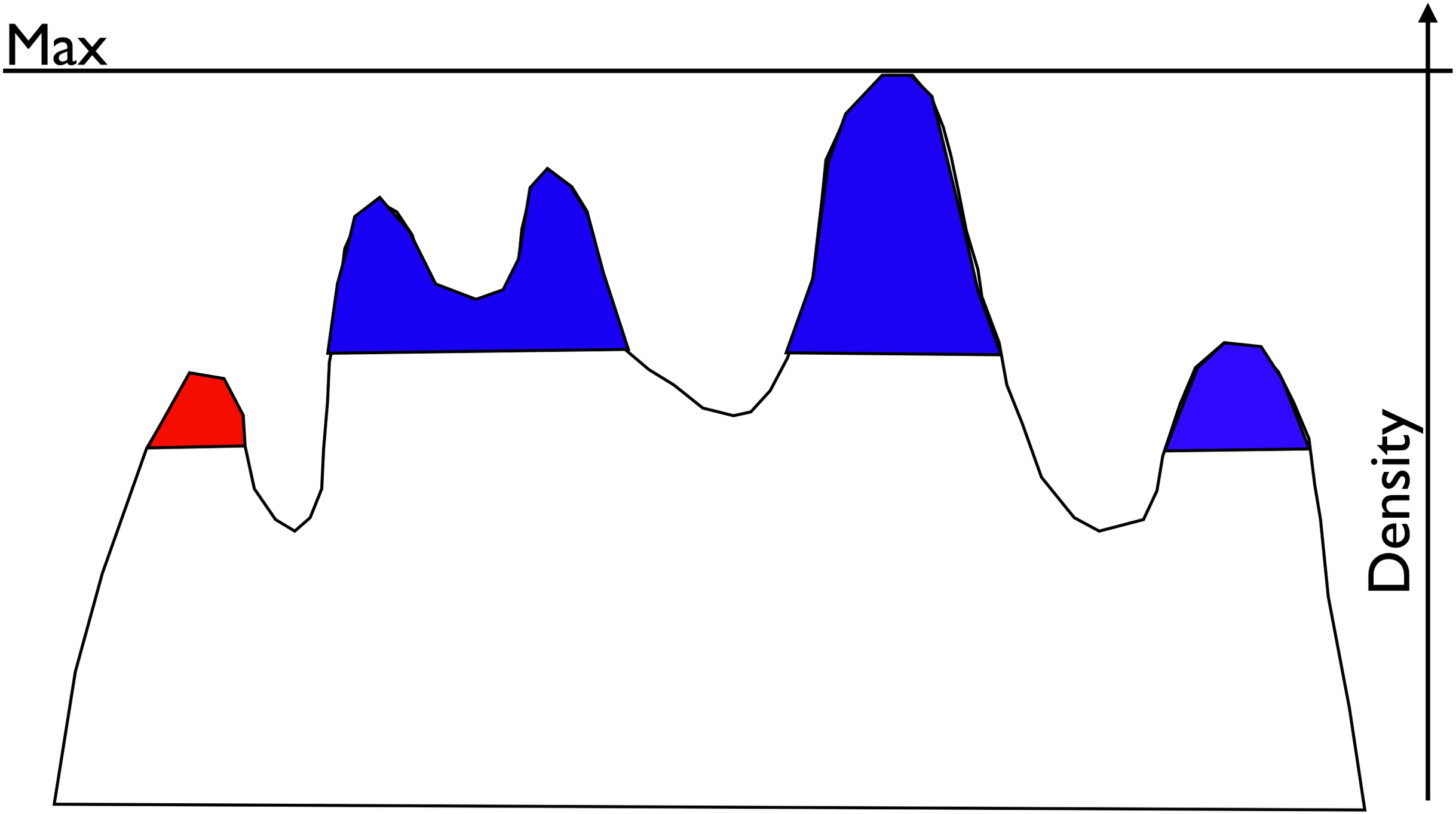
Finding Clumps



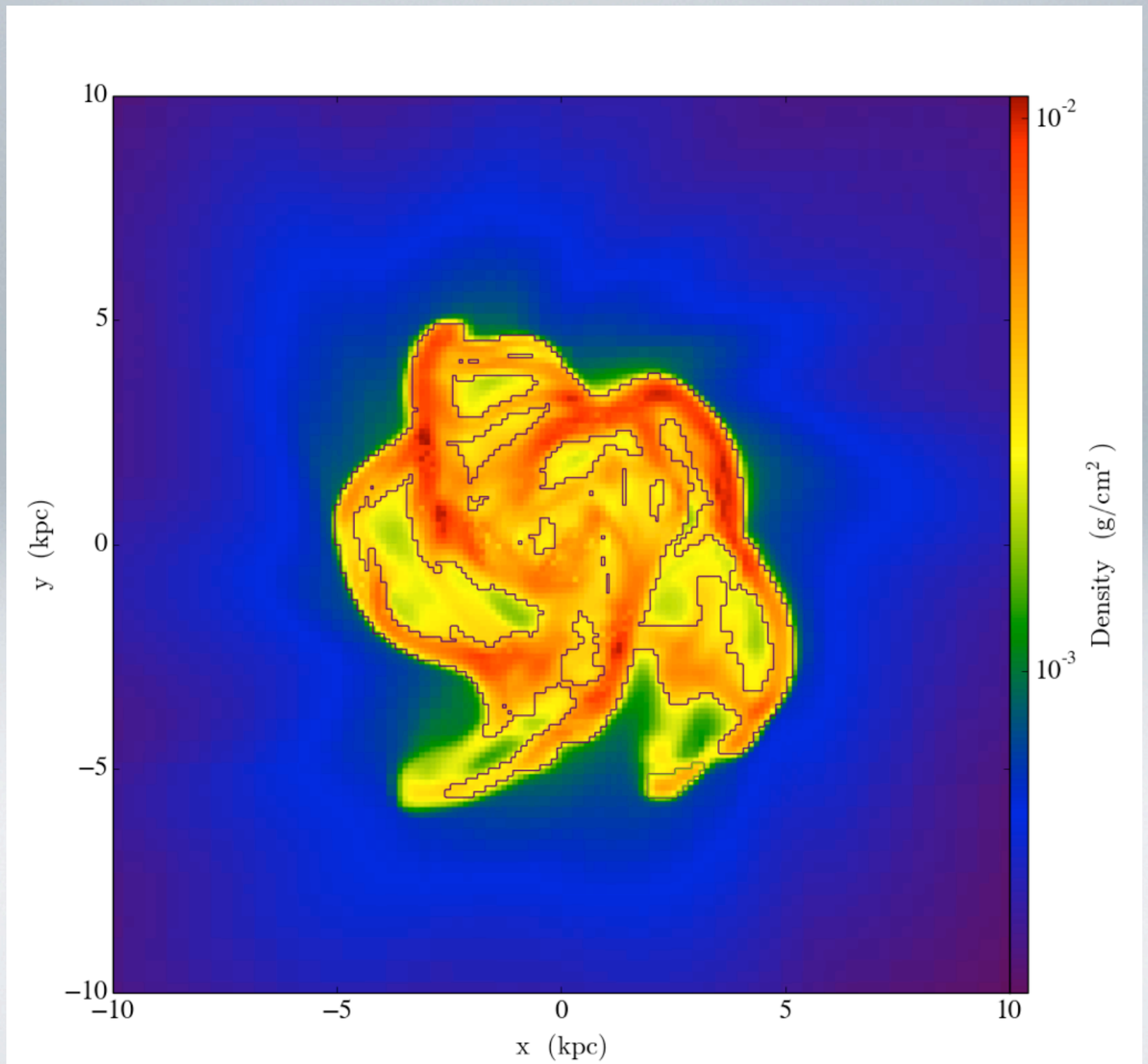
Finding Clumps



Finding Clumps



Clump Finding - "12_find_clumps.py"



Clump Finding - “12_find_clumps.py”

```
yt : [INFO     ] 2013-10-18 12:14:37,932 Getting field x from 1
yt : [INFO     ] 2013-10-18 12:14:37,932 Getting field y from 1
yt : [INFO     ] 2013-10-18 12:14:37,933 Getting field dx from 1
yt : [INFO     ] 2013-10-18 12:14:37,933 Getting field dy from 1
yt : [INFO     ] 2013-10-18 12:14:38,015 Saving plot clumps_Projection_z_Density.png
Clump mass: [4926399.7741416385] Msun
Clump mass: [663288412.58299804] Msun
```

“cat galaxy0030_clumps.txt”

```
Clump:
Cells: 38
Mass: 4.926400e+06 Msolar
Max grid level: 8
Min density: 3.203149e-24 g cm^-3
Max density: 7.243956e-24 g cm^-3

Clump:
Cells: 4912
Mass: 6.632884e+08 Msolar
Max grid level: 8
Min density: 3.163817e-24 g cm^-3
Max density: 7.733779e-24 g cm^-3
```

Clump Finding - “find_clumps_big.py”